SMARTPLANTER

A white paper submitted in fulfillment of the contest requirements for

the

Cornelius van Drebbel's

Mad Design Contest

Hardware Submission KORE9814

Co-Sponsored by

NEC Electronics America

&

EDN Magazine

Submission Date: January 5, 2005

by

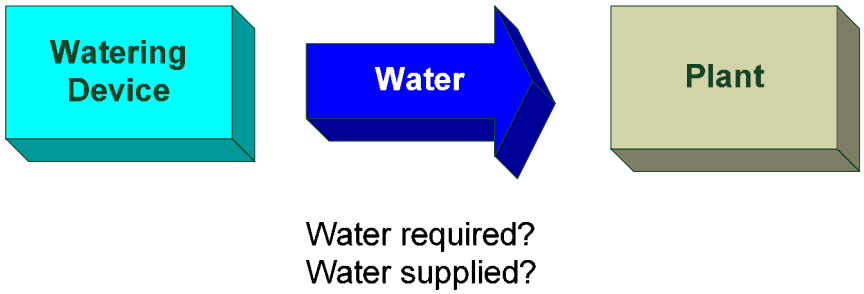
Alain d'Espaignet

INTRODUCTION

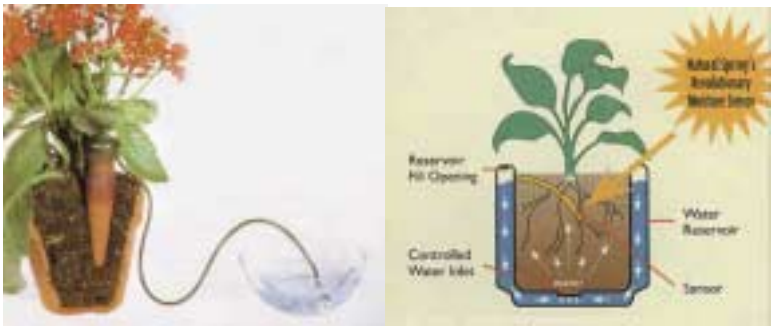
This paper presents a practical application of the NEC KORE9148 hardware kit in the implementation of an automated plant care system. Heretofore, called the *SmartPlanter,* it is a closed loop system that measures and controls parameters essential to keeping plants healthy. It uses a simplified evapotranspiration(ET) model based on the FAO-56 Penman-Monteith equation and physical measurements, to estimate the watering frequency and duration required to replenish water losses and then used to actuate a watering pump. The *SmartPlanter* keeps plants healthy and fertilized at markedly reduced maintenance. Actual soil moisture measurement closes the control loop and is used to continuously improve the ET model. Seasonal model adjustments are made based on a real-time clock/calendar. Configuration and status is through a keypad/LCD user interface. Low water supply level activates audible and visible alarms. Solar power keeps a LiIon battery charged, storing energy to operate fertilizer dosing and watering actuators. Low maintenance with improved plant health/growth is achieved through a unique combination of software and hardware.

Application description and purpose

Several automated watering devices are on the market today but most of the affordable ones are open-loop devices. They feed water to plants with no knowledge of how much water is currently required by the plant or how much watering took place.



Open loop systems tend to over water or under-water resulting in reduced plant growth and health.



Over-watering causes

- Root rot

- Encourages fungus growth

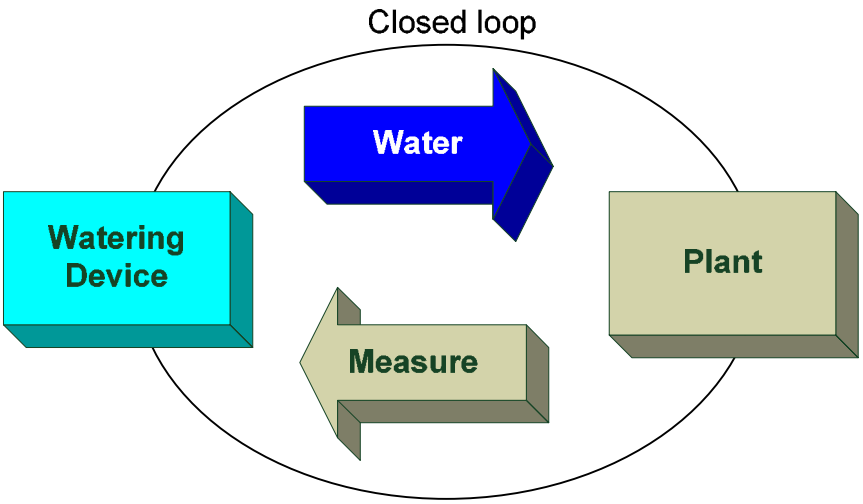
- Eventual plant loss under prolonged extreme circumstances

Under watering causes

- Wilting & shock

- Plant death under prolonged extremes

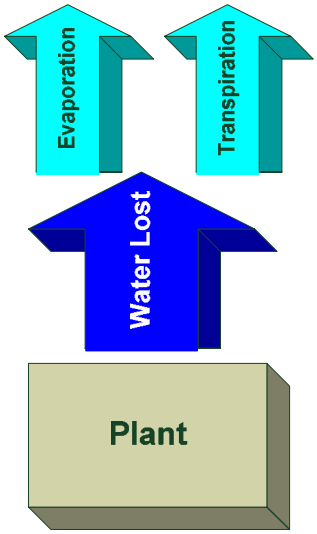
There are occasions when it is desirable to maintain optimal plant growth and health – such as in high-priced horticulture – and then a closed-loop system is often preferred. A closed -loop system predicts how much water plants require and measures the effects of watering, correcting itself to improve future predictions.



Plants loose water through two main processes:

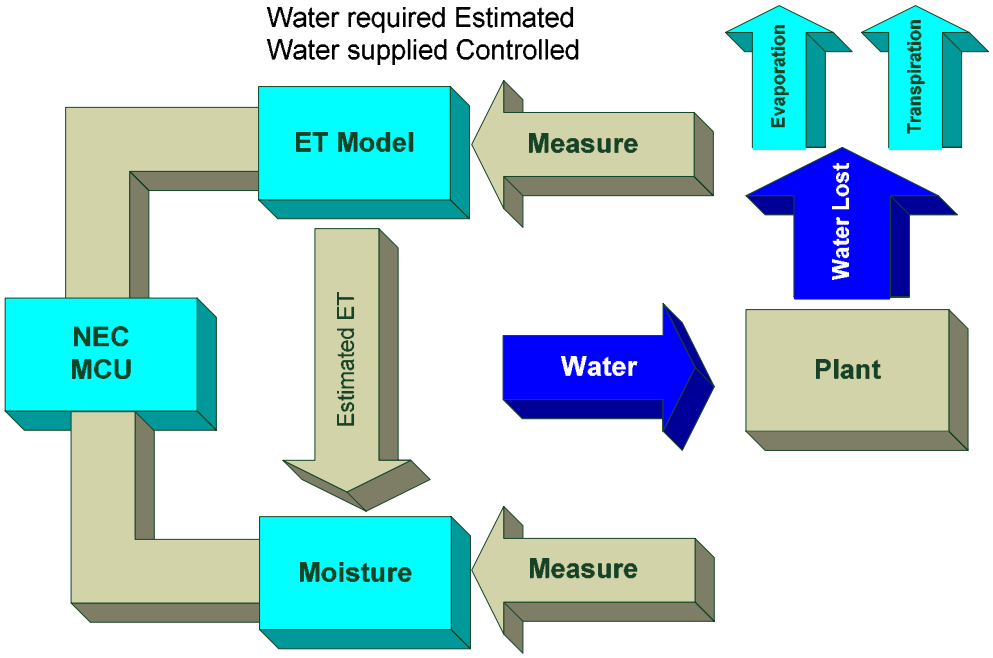
- **Evaporation** at the soil surface.

- **Transpiration** of the plant itself through its foliage.



The term evapotranspiration (ET) is often used to describe the combined effect. ET represents the water lost by a plant. Replenishing the lost amount through watering offsets the effect and keeps the plant under optimum conditions. A mathematical model for ET was developed by Penman in 1965. It was subsequently simplified to what is known as the FAO-56 Penman-Monteith equation. The model still requires several measured parameters for the computation of ET and generally would use data from a weather station. In this implementation some assumptions are made by categorizing the plant's micro-climate to simplify terms of the equation. The simplified model uses only the following measurements: solar irradiation, humidity, temperature and dew point. To compensate for soil type variations, plant type and other unknown factors, the SmartPlanter, uses a feedback loop to make corrections to the model.

The SmartPlanter is based on the NEC MCU to perform the following main functions: Measures the following: Humidity, Temperature, Dew point, and Solar irradiation These measurements are used to estimate the EvapoTranspiration (ET). The ET estimate is used to control the watering duration and hence the amount of water dispensed to the plant.



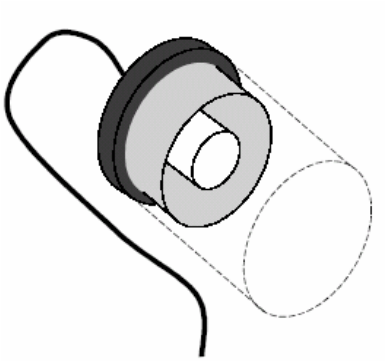
The system also measures Soil Moisture to verify and correct the ET model. The SmartPlanter functions primarily as a moisture controller. Other functions include monitoring and controlling nutrients and logging data for analysis.

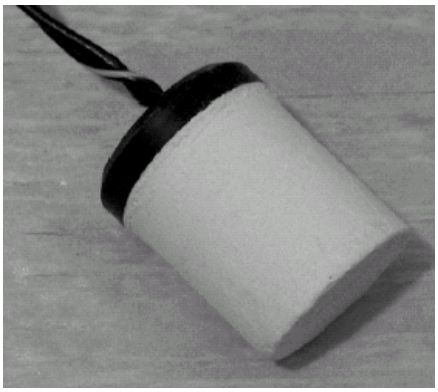
SOIL MOISTURE MEASUREMENT

There are many techniques used to measure soil moisture. To keep the SmartPlanter affordable the granular matrix sensor technique is used. The most affordable form of such sensors is the Gypsum block. Gypsum blocks have been used in soil science for 50 years (Bouyoucos, 1965) to estimate soil moisture and their performance is well documented. Gypsum blocks actually do not measure soil moisture directly; instead they measure soil tension, which is related to the amount of water that a plant/soil combination can absorb water. This measurement is often referred to as soil suction or pressure, the ability of the soil to draw water. The units are thus expressed in Kilo Pascal.

What makes this technique of soil measurement attractive is the relatively low cost of the sensor and the simplicity of the associated measuring circuit. In addition moisture measurement is independent of soil type.

Gypsum block soil moisture sensor construction

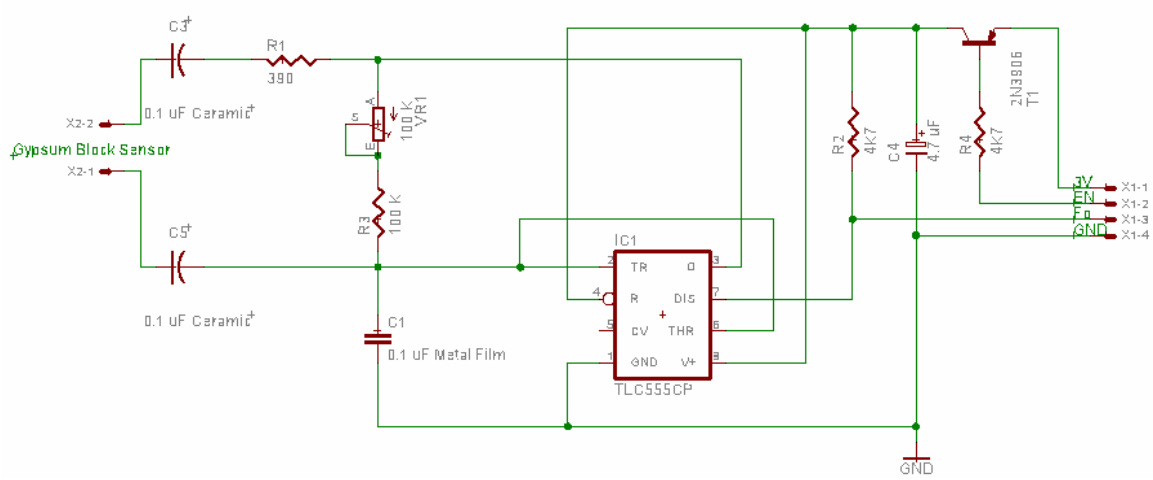




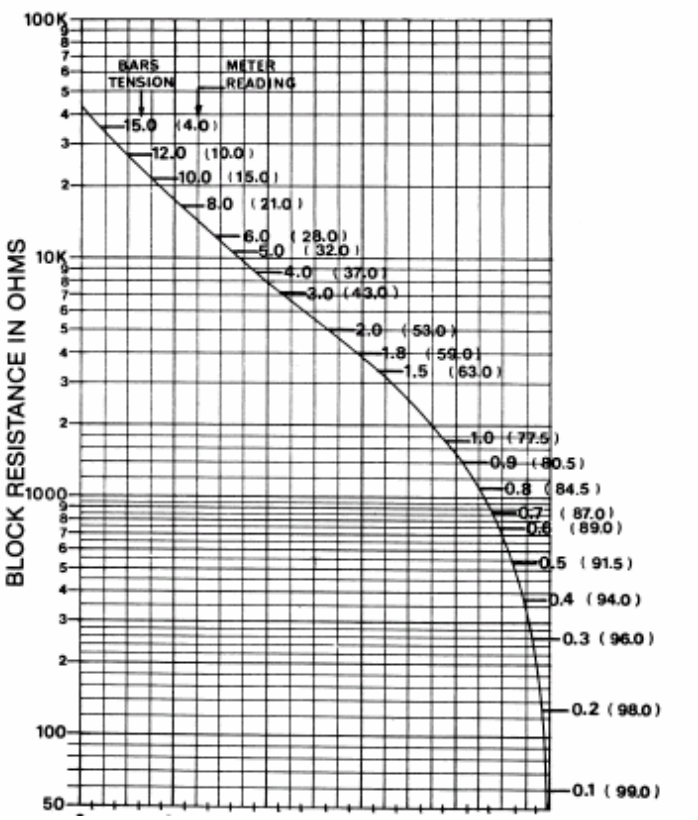
Gypsum blocks are cylindrical in construction with a diameter of 25 mm and a length of 35 mm. Two concentric rings of wire mesh are embedded in the gypsum block and make up the electrodes thus confining the flow of electric current between the electrodes to the interior of the block, eliminating the effects of soil conductivity. Conductivity is not a true measure of moisture as it varies with PH and its effects are not desired. The gypsum acts as a buffer against the effect that salts present in the soil might have on the electrical conductivity. A controlled mixture of gypsum or Calcium Carbonate, $CaSO_4$ (also known as Plaster of Paris) is placed between the electrodes. The resistance between the electrodes varies with the moisture content of the gypsum block, which in turn varies with the suction pressure in the surrounding soil. Measuring the AC resistance of the gypsum blocks provides information about the suction pressure in the surrounding soil. The measured AC resistance varies as a function of temperature but equations do exist to compensate of the effect of temperature dependence.

Gypsum block soil moisture measurement

To measure soil moisture using a gypsum block one has to measures the AC resistance between the block's electrodes. Applying DC currents to gypsum blocks causes gasses to form at the electrodes hence an AC free of any DC bias current is required. The requirements are that the block should be excited with an AC voltage source of around 1V peak to peak. Care should be taken not to maintain this driving voltage for long periods of time. Short bursts of bipolar voltage pulses of a few volts in amplitude should suffice to give accurate and repeatable readings. There are many ways in which these requirements can be met but the simple circuit below is the most economical and uses very little power.



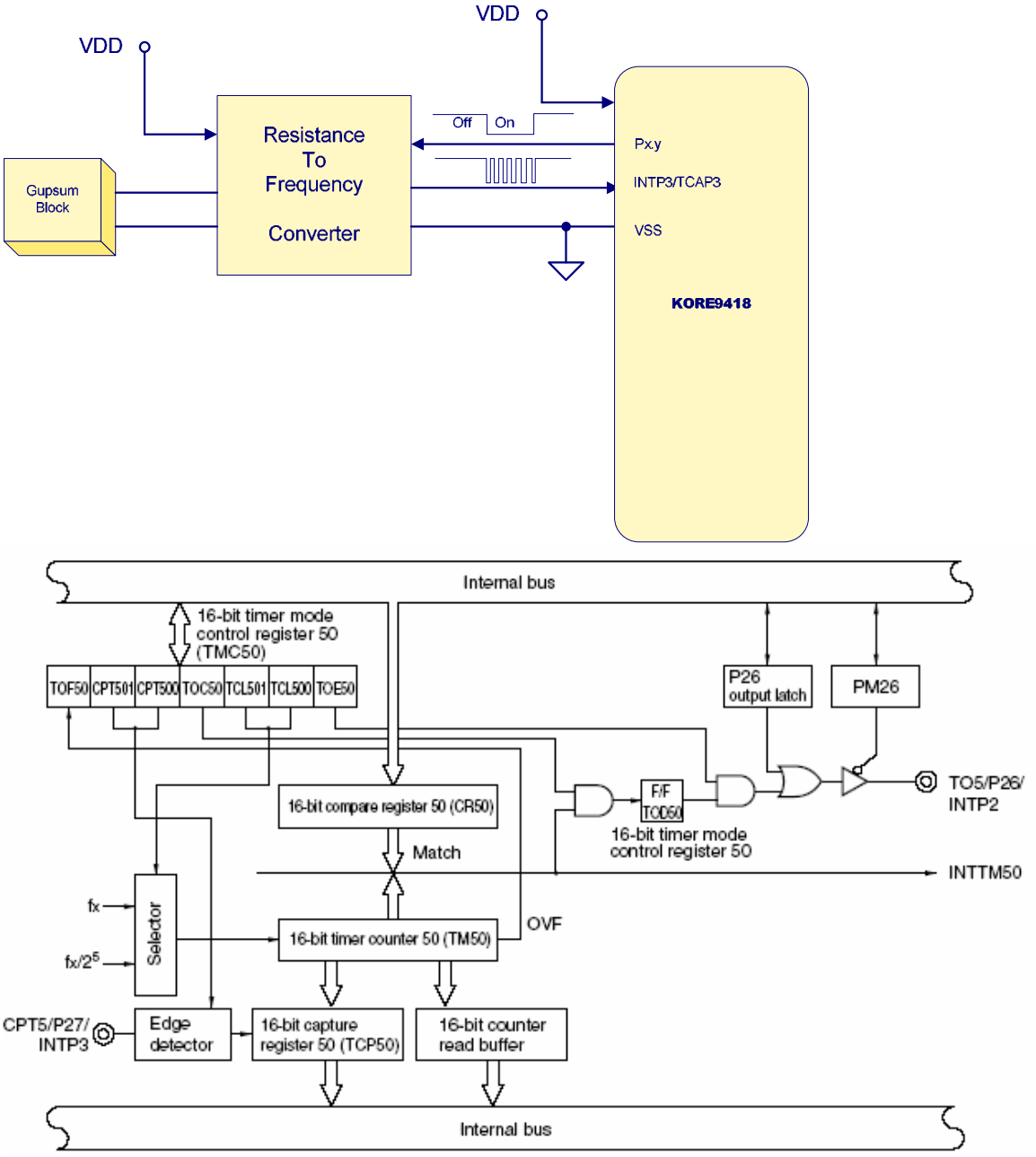
In the circuit above the gypsum block sensor's resistance is part of an RC timing network. As the resistance of the gypsum block varies with soil tension the output frequency of the circuit varies. The output of this circuit is a square wave with frequency proportional to the block's resistance and hence to the soil tension. Experiments have shown that the output frequency varies between 28 Hz when the block is dry to a maximum of around 1.3 KHz when wet (immersed in water). The circuit can be considered as a resistance to frequency converter. A typical gypsum block resistance curve as a function of soil tension is as shown below:



Note the S shape of the curve's relationship. This implies that at the extremes that is extreme dryness or wetness it the variations in resistance are the smallest. Although the block's resistance can be measured using strictly analog techniques using an AC bridge followed by a precision rectifier and filter feeding to an A/D converter, the resistance to frequency approach offers better resolution and has a dynamic range of slightly more than 1000. Measuring soil tension is hence a matter of accurately measuring the frequency or period of the output pulse train of the circuit above.

Period measurement technique

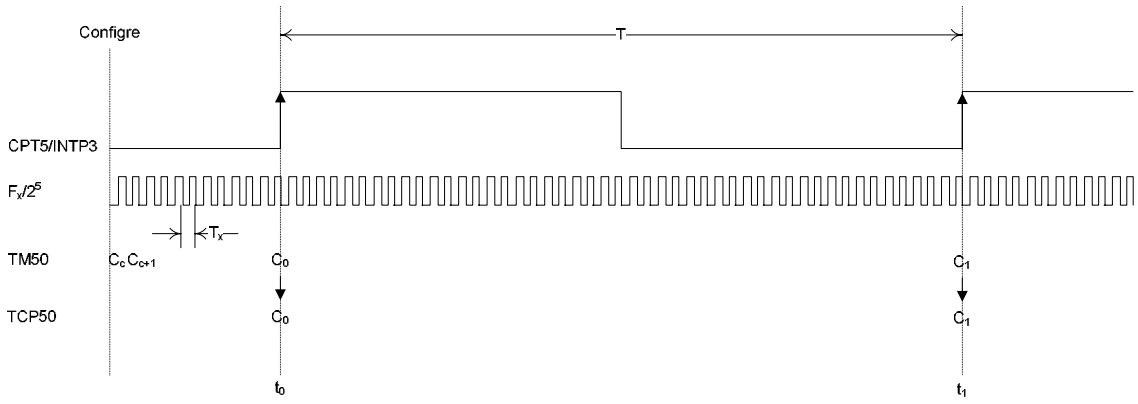
The NEC uPD78F9418's 16-bit timer with capture register, driven by an external edge detector as trigger is ideally suited for this type of measurement.



The gypsum sensor "resistance-to-frequency" circuit's output is connected to the 16 bit timer T50 capture input CPT5. The edge detector on the CPT5 input is configured to trigger a capture on rising edges. T50's clock source selector (CPT501/CPT50) is configured to use $f_x/2^5$ as the clock source driving the 16 bit counter TM50. The capture

function does not work at the higher clock rate options. In this configuration the T50 counter is free-running and increments its count by one every $1/f_x/2^5$ seconds. Whenever a rising edge is detected on CPT5 the current 16-bit counter value (TM50) is captured and latched in the capture register TCP50. Note that the sensor's frequency output is also connected to the INTP3 interrupt.

To measure a pulse train frequency one simply has to measure the time elapsed between two subsequent edges of the pulse train period using an accurate and known time base as reference. In this case the $f_x/2^5$ provides time base source where f_x originates from an external 4.9152 MHz crystal.



At the time that timer 50 is configured TM50's count is C_c. It then counts cycles from its configured clock source of $f_x/2^5$ as C_{c+1}, C_{c+2} and so on, in a free-running fashion until it overflows. It then rolls over and continues counting from 0 and up. At t_0, when a rising edge is presented to the CPT5 input, the TM50 count of C_0 is captured and latched into the capture register TCP50. On the subsequent rising edge at t_1, TM50 value is captured as C_1.

To obtain the frequency (F) or period (T) of the unknown signal presented at the CPT5 input the following applies, assuming that $C_1 > C_0$, that is that TM50 has not overflowed.

Since F_x = 4.9152 MHz (on the KORE9418)

$T_x = 1 / F_x =$ 20.345 us

Period T = $(C_1 - C_0)\ T_x = 20.345\ (C_1 - C_0)$ us

Frequency of measured signal F = 1 / T

This technique works as long as the dynamic range of F can be resolved with the chosen F_x and the 16 bit limitation of T50. $F_x/2^5$ has to be at least 2 times F_{min} but the higher the better and the better the resolution. Unfortunately we have limited control on the timer's clock source since the capture function is limited to $F_x/2^5$, in any case as shown below this is more than adequate for the frequencies being measured.

As a sanity check assume that the expected frequency range of the signal to be measured is between 28 Hz and 1.3 KHz as observed in preliminary tests.

The highest timer 50 count occurs at the lowest frequency. This count should not exceed the maximum count possible with 16 bits i.e. $2^{16} - 1$ or 65535. At 28 Hz, T = 0.035714 seconds which is equivalent to a count of pulses of $T/T_x = 1755$ which is less than 65535.

Similarly the lowest timer 50 count occurs at the highest frequency of 1.3 KHz. In this case T/Tx = 37 counts.

Hence it clear that it is feasible to measure soil tension by indirectly measuring the sensor's period T. The resolution in the sensor's period measurement is 20.345 µs.

The stability of the measurements is dependent on that of the external crystal. Other variations in measurements are due to the temperature dependence of the timing capacitors used in the measuring circuit.

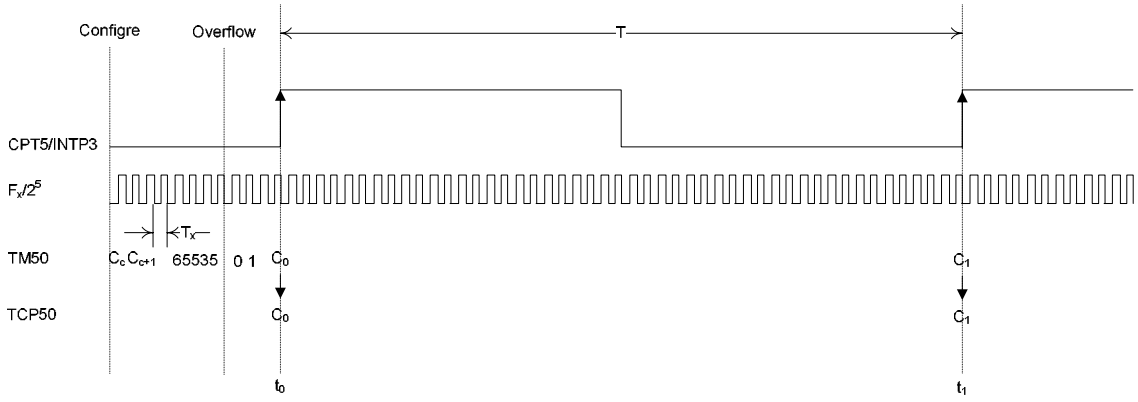
Special cases

All is fine except for one complication. Timer 50 may overflow and one can no longer assume that $C_1 > C_0$. Fortunately this condition can be detected using the overflow register TOF50. Three conditions may arise:

1. TM50 overflows both before time t_0 and t_1
2. TM50 overflows once before t_0
3. TM50 overflows once after t_0 but before t_1

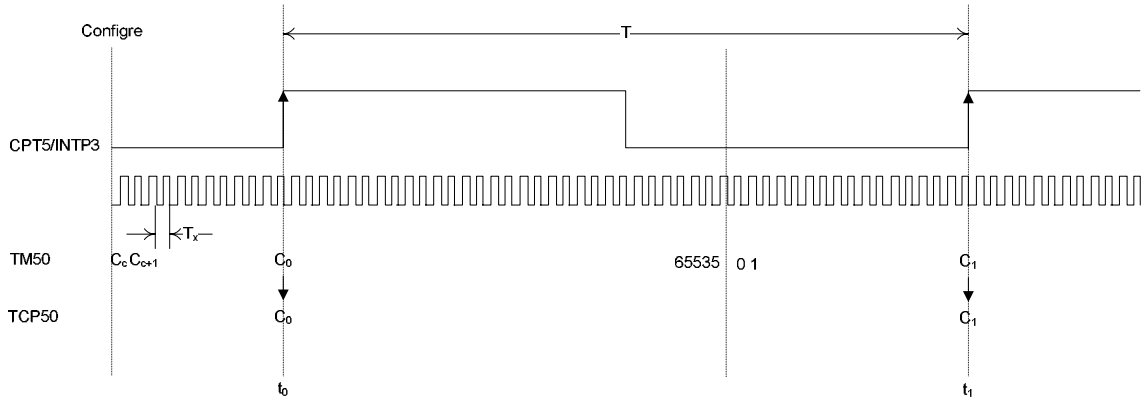
With the knowledge that the expected frequency range of the sensor is between 28 Hz and 1.3 KHz it is clear that condition 1 should never occur. This leaves us with having to compensate for conditions 2 and 3 only.

Condition 2



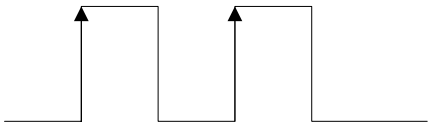
In condition 2, an overflow occurs before the first rising edge. It is cleat that given the expected count range, a second overflow cannot occur, hence $C_1 > C_0$. All we have to do is clear the overflow flag.

Condition 3

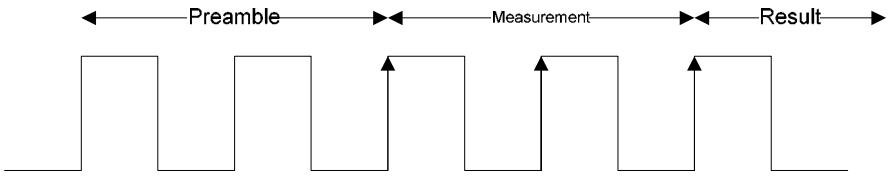


In condition 3, an overflow occurs between times t1 and t0 as shown above. The corrected count is thus: $(65535 - C_0) + C_1$ and not $C_1 - C_0$ as in the absence of any overflows. We note that an overflow condition occurred account for it in calculations and clear the overflow flag.

The software implementation of the above technique relies on the use of the INTP3 interrupt to detect edges from the pulse train to be measured. A port is also used to turn the sensor circuit on and off. The sensor circuit is kept off at all times to keep power consumption low. It is turned on only when measurements are to be taken.



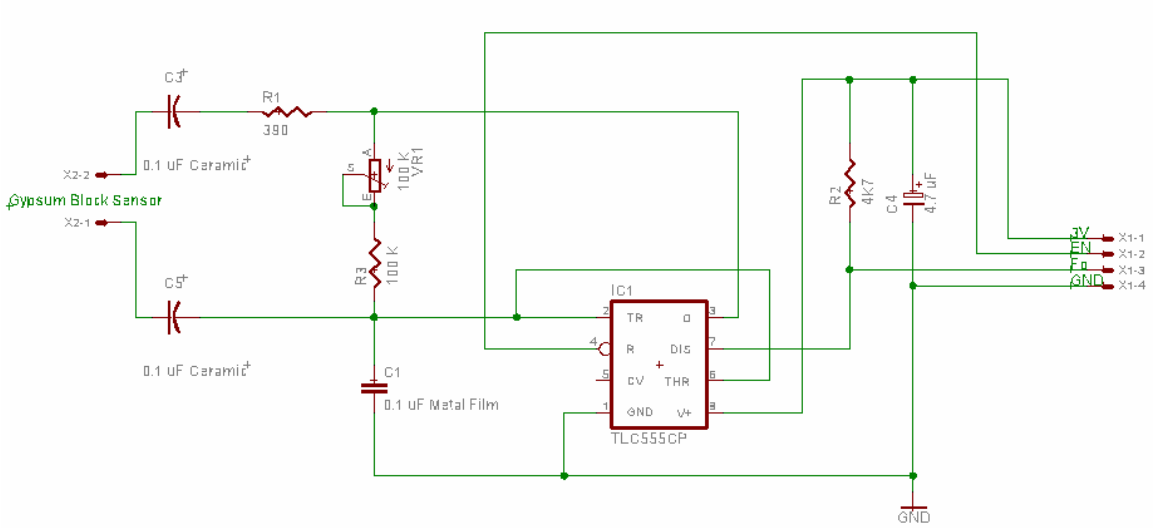
It is possible to excite the sensor with exactly two periods (two rising edges) of the bipolar waveform. This is achieved by counting the edges detected on INTP3 and turning the measuring circuit off immediately after the second detected rising edge.

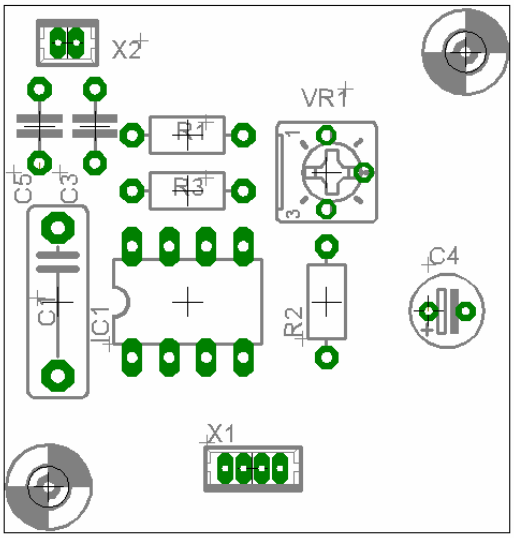


Experience has shown that it is best to actually let the moisture measuring circuit run for a few cycles to let it stabilize before taking measurements. The duration of the preamble time should be greater than the maximum expected period of the signal. Two edges are detected to take measurements of C_0 and C_1. On the third detected edge C_0/C_1 is adjusted for any overflows from which the final result can be calculated from knowledge of F_x.

To conserve power the supply to the moisture measuring circuit was initially fed through a PNP transistor controlled by a port. This approach turned out to be overkill since the low power CMOS version of the LM555, the TLC55 was being used. The TLC555 draws a few micro amps when it is not driving a load. Instead the RESET input controlled from a port is used to start/interrupt the pulses to the gypsum block sensor. It is critical not to keep the sensor continuously energized because of galvanometric effects.

The final moisture sensing circuit used is as shown below:





Moisture measurement software

The following program takes moisture measurements every 10 seconds and sends the results to the serial port for logging and graphing. Accent is placed on moisture measurement aspects.

```
[code removed for publicatoin]
```

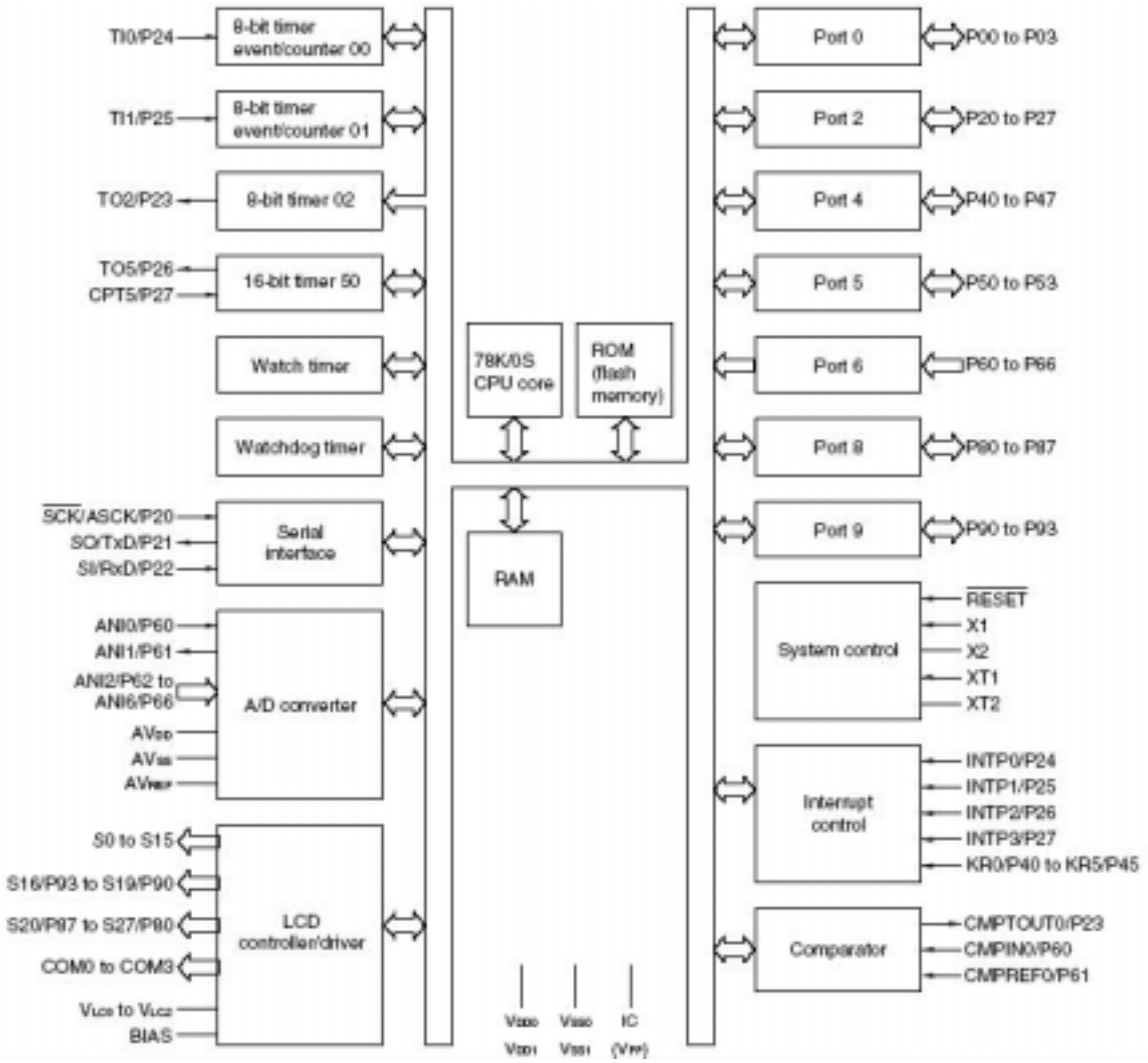
CHAPTER II: DESIGN CONSIDERATIONS

- The SmartPlanter should primarily be solar-powered and use a battery for back-up purposes.
- The SmartPlanter should be low-cost, problem free and easy to operate.
- It should use as many features of the NEC upd78F9418 as possible.

CHAPTER III: SPECIFIC FUNCTIONS USED IN THE MICROCONTROLLIER

Hardware function usage

Below is a block diagram of the updPD78F418. The table that follow show what features of the microcontroller are used in this application.

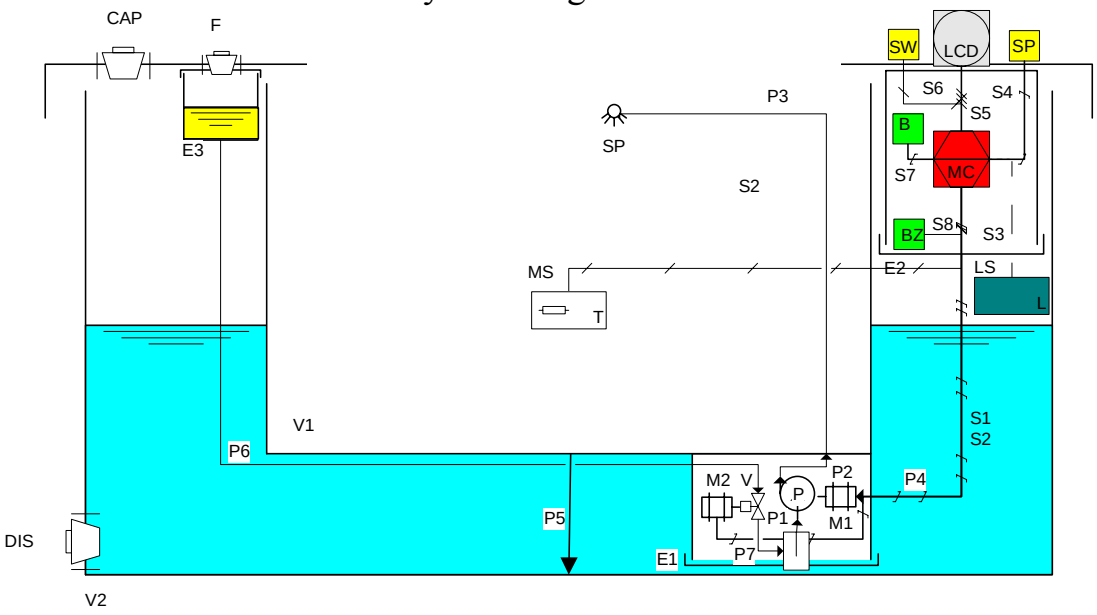


Features	Actual Use	Intended Use
8-bit timer event/counter 00		Used for accurate delays
8-bit timer event/counter 01		Used to drive the watering pump in PWM mode
8-bit timer 02	Using square wave mode to drive the buzzer tones	
16-bit timer 50	Used to measure frequency output of the moisture sensor	
Watch timer	Used to keep a real-time clock calendar	
Watchdog timer		To recover from software crashes
Serial Interface	Used to send log data to host computer	
A/D Converter	Used to measure voltages, namely…	
LCD controller/driver	Used to drive a 4-digit 7-segment LCD display in static mode	
Port 0		Used to drive the keypad
Port 2		
P2.0 SCK		
P2.1 SO/TXD		RS232 transmit
P2.2 SI/RXD		RS232 receive
P2.3 CMPTOUT0/T02	Used to drive the buzzer	
P2.4 INTP0/TI0	Used to enable/disable the moisture sensor	
P2.5 INTP1/TI1		SHT11 DATA
P2.6 INTP2/T05		SHT11 EN
P2.7 INTP3/CPT5		SHT11 CLK
Port 4		Keypad interface
Port 5	Used to control actuators	
Port 6	Used to measure various voltages	
Port 8	Used to drive LCD display	
Port 9	Used to drive LCD display	
Comparator		To sense solar power level

The ‘Actual Use’ column in the above table represents the functions that are already implemented in the design. The ‘Intended Use’ column shows functions I based my design around, but ran out of time to actually implement them.

CHAPTER IV: COMPREHENSIVE BLOCK DIAGRAM OF ALL MAJOR FUNCTIONS AND CONNECTIONS

System diagram



Description (of system diagram)

Water storage

Vessels **V1** and **V2** form a water storage container of a few gallons. The container is covered with a lid to reduce water evaporation.

Water refill, discharge and drainage

CAP gives access to fill the container and **DIS** is a discharge valve used to drain water out for maintenance. **V1** and **V2** are inter-connected by drain pipe **P5** allowing water seeping through the plant soil contained in **V1** to evacuate. A tray is optional on the outside especially for indoor plants to keep drain water contained.

Enclosures

Within the V1/V2 container are three separate enclosures: E1 Actuators, E2 Controller and E3 Fertilizer storage respectively.

Actuators

E1 contains the watering and fertilizing actuators. Fertilizing is done by mixing controlled amounts of fertilizers in with the watering system. Watering is through magnetically coupled pump P operated by motor M1.

Fertilizer storage and mixing

Liquid fertilizer is stored in enclosure E3 and is refilled through access cap F. Fertilizing is controlled by pinch valve V activated by motor M2. Fertilizer flows from E3 through pipe P6 and valve V to the mixing point reached via pipe P7.

Watering

The water intake for the pump is the V1/V2 container via P1 and the outlet is through pipe P3 leading to interchangeable sprinkler head SP. SP can be sprinkler or mister heads selected for maximum plant coverage.

E1 signals

S1 and S2 carry fertilizer and pump motor control signals from the controller enclosure E2 running through sealed pipe P4 to enclosure E1.

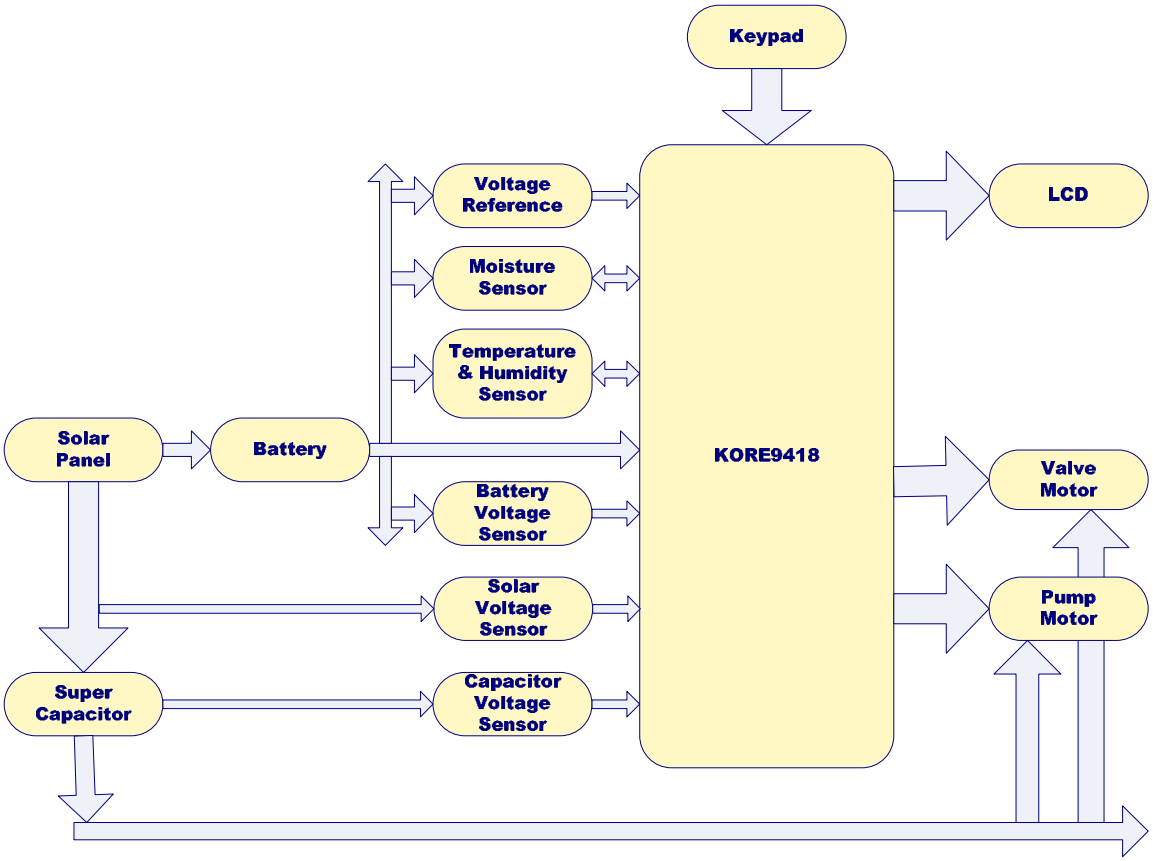
E2

Enclosure E2 contains a micro controller MC, a LiIon battery B and an audible buzzer BZ. A user interface is provided through a liquid crystal display panel LCD and a keypad SW.

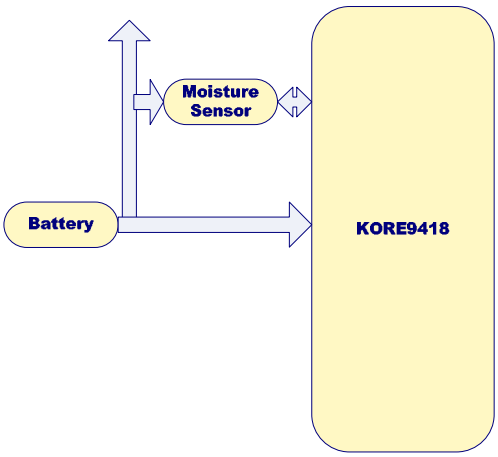
Solar power

The unit is solar powered via solar panel SP.

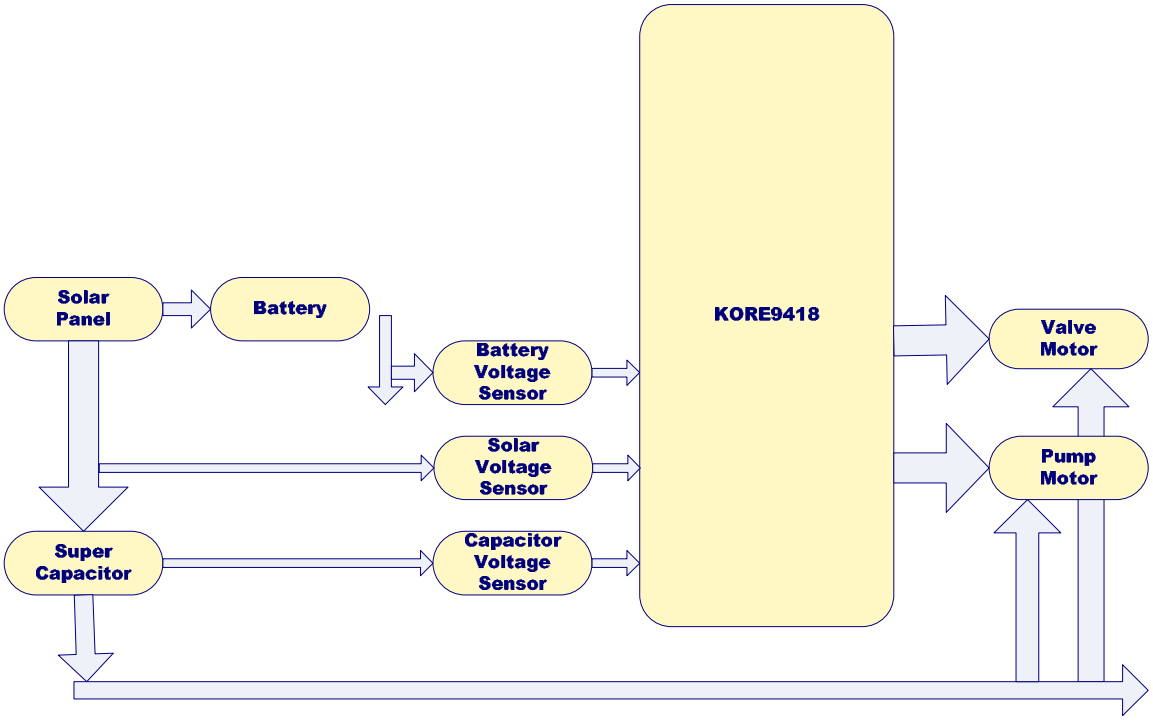
Controller (block) Diagram



Moisture Sensor Block Diagram



Power Sensing and Actuator Block Diagram

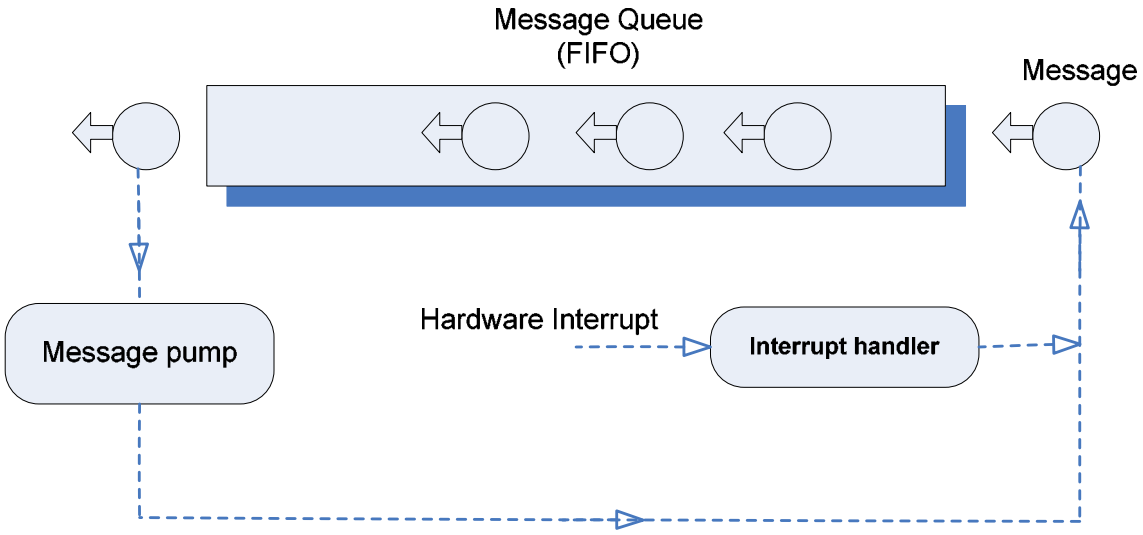


The message pump paradigm

Perhaps I have been programming under the Windows operating system for too long and have fallen in love with the concept of message pumps. But, I find that it has a tendency to simplify the software in both real-time systems and in embedded systems. Therefore I decided to implement a subset of this software technique for the upd78F9418.

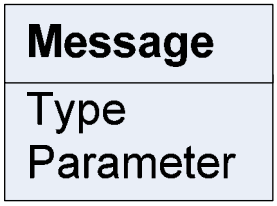
The message queue paradigm simplifies and clarifies software in the way that it decouples the originator of an event to the code processing the event. Too often I have seen complicated interrupt handlers that do too much and communicate state information with the main loop through flags. Surely there is a more elegant approach. Just because it is at the bare silicon level does not mean that it has to be quick and dirty. Embedded software can be elegant and simple to maintain too.

The idea is based on a queue or FIFO data structure. Messages are entered in the queue at one end and removed at the other end. Code that inserts messages in the queue should never remove them from the queue for simplicity. Typically, code that en-queue messages are interrupt handlers and code that de-queues messages are in the main program's processing loop.



Messages originate from events. These are generally true real-world hardware events such as a limit switch interrupt but they can be software events too, such as to convey a change of state in the scope of a program. Messages can carry parameter values as well.

In the Windows world a message has a type, generally an enumeration or #define with a name such as WM_CLOSE. Bbut it is also capable of carrying two parameters, a 16-bit integer as WPARAM or a 32-bit integer LPARAM. LPARAMS can represent pointers hence a message can in principle carry with it a pointer to a huge data structure. In this reduced implementation a message only has a type and a single parameter.



The definition for a message, message type and parameter are as follows. Here are the full contents of the MessageQueue.h header file.

```
[code removed for publication]
```

To work with a message queue is as simple as working with a FIFO data structure with just one complication. Code that en-queues and de-queues messages should never step over each other to make the state of the queue inconsistent. Therefore we need a way to synchronize access to the queue variables. This can be done with a semaphore or flag that should be used by only one piece of code at any time. Here is the code.

```
[code removed for publication]
```

This is all there is to it. Now observe how this device simplifies and makes things clean.

In this usage sample I want to repeatedly take sample readings every 10 seconds, or every hour - the duration does not matter. Here is a typical usage of the message queue and I use it in most of my time-based applications but it can also be used to queue in keypad entries in the queue by a keypad interrupt handler for processing by some main processing loop.

Here is the queue in action. This sample sends the time and date to the serial port every 10 seconds.

```
[code removed for publication]
```

Here is the real-time clock module:

This is the header file, watch.h

```
[code removed for publication]
```

This is watch.c

```
[code removed for publication]
```

This is the piece that deals with the watch timer and inserts a MSG_SAMPLE type message in the queue at the sample rate.

Here is the short and sweet header file rtcq.h

```
[code removed for publication]
```

And here is the implementation file rtcq.c

```
[code removed for publication]
```

The important bit is the watch timer interrupt handler WATCH_ISR.

It simply counts the number of ½ second ticks since this is the way the watch timer works. This handler is called every ½ second. To count in seconds we simply count the number of half seconds.

If (g_sruiHalfSeconds % 2) == 0) is true on every second. This gives us an if statement where the body executes every second. On every second we update the real-time clock and calendar data structures via a call to update_watch(). We then check if we have reached the specified sample rate in seconds and generate a message to the queue. This makes the call to GetMessage(…) in main succeed with the message type MSG_SAMPLE.

Like I said earlier, other types of events can be generated in a similar fashion by inserting messages of different types in the queue. This technique is also useful for handling interrupt driven serial communications. A serial receive interrupt handler routine would buffer character received until an end of message character is received at which point it may signal to the main loop the arrival of a complete message and its type by simply making an entry in the message queue. In this way the main code is not littered with serial communications related variables and handles things at a slightly higher level. This message queue technique is used in the SmartPlanter code to implement state changes and direct the main loop to specific state processing code sections in a clean and readable way.

The serial port

This application does not really call for a serial port. However, the latter proved invaluable for debugging. For those who are interested in studying the behavior of plant species it is advantageous to have the capability to log data to a host computer. The serial port is meant to be an optional feature for those that require it. In the final product idea that this prototype represents, this optional feature is meant to be complemented with adequate host software. The software should provide the ability to send specific commands to the planter to request and/or set parameter values and for logging and graphing measurements.

The serial port on the upd78F9418 is easy to use. Unlike the 8051 family of microcontrollers it uses a dedicated baud rate generator clock instead of reusing generic timers. This makes configuring and using it less complicated. The serial port is used in a number of programs presented here. They all make use of the following functions found in serial.h and serial.c. The serial interface on the M-Station and the associate MSTTERM software is useful however since MSTTERM does not seem to want to remember the baud rate last setting, I found it annoying having to set it up every time I used the KORE9418 at a baud rate other than the default of 9600. I preferred higher rates but ended up using mostly 9600 because of this annoying thing about MSTTERM. This is reflected in the serial software.

Serial.h

```
[code removed for publication]
```

Serial.c

```
[code removed for publication]
```

Liquid Crystal Display Driver

This application definitely calls for a nice graphic LCD display, perhaps even a custom designed LCD with icons to help users select plant species and other configuration parameters.

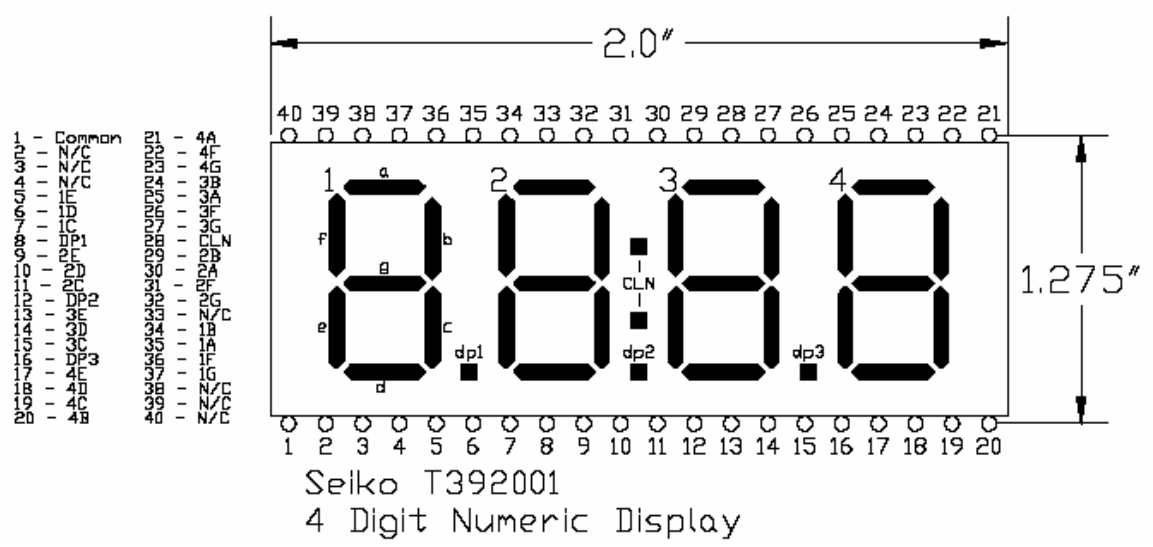
Unfortunately, by the time I heard about the Mad Design Contest and created an application, all of the graphic LCD-based development boards had already been spoken for. So, I used the 7-segment LCD of the upd78F9418 and made do with it.

Finding low cost 7-segment LCD modules in unit quantities is not an easy task. I eventually honed in on two offerings. One was a 4-digit LCD available from DigiKey and the other a 4-digit 7-segment LCD found at an online electronic surplus store at a mere $4 each. Four digits are the best I could find in small quantities and I planned to use the most digits possible.

The upd78F9418 is capable of driving a maximum of 3 digits and associated decimal points in the static mode. I dropped the use of the decimal points in favor of more digits.

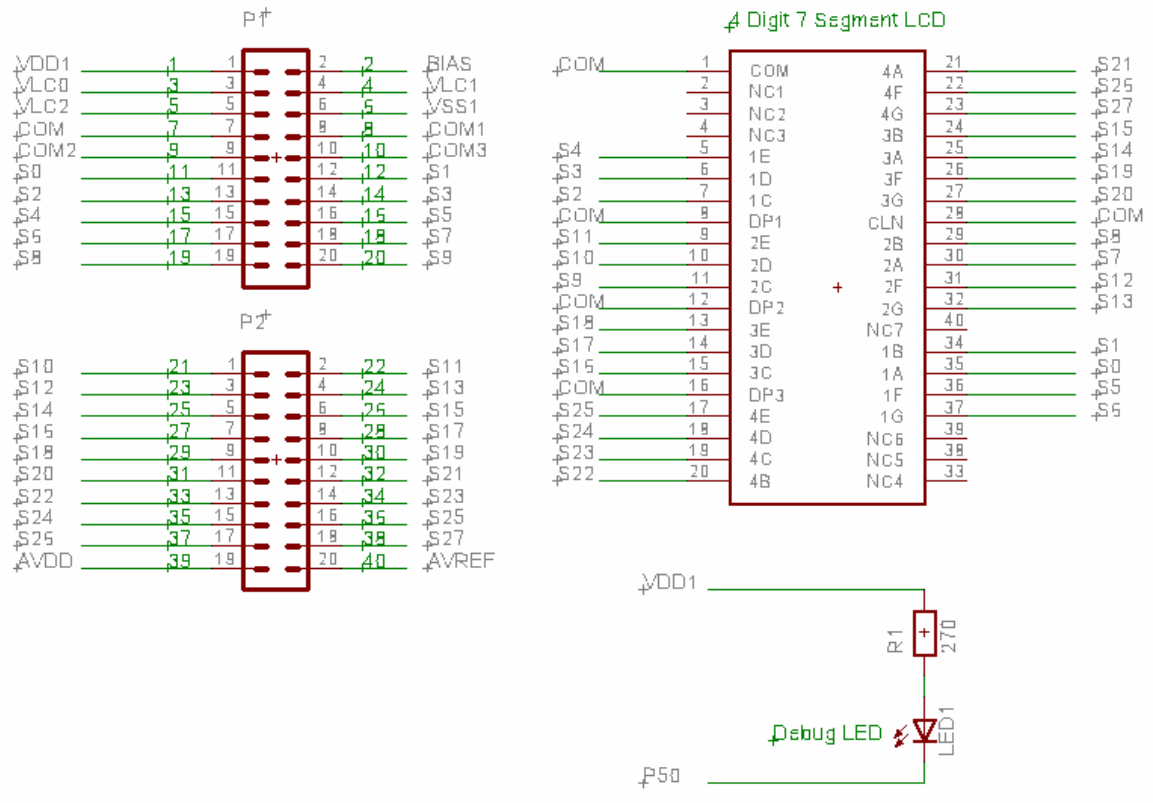
As I worked with the display the lack of decimal points became more and more unnerving and eventually I resorted to an electronic trick to drive decimal points and the colon symbol from other ports. This trick only works in the static mode letting me have 4 full 7-segment digits and the decimal points and colon symbol for the time display.

The Seiko T392001 available from www.eio.com is the 7-segment LCD I chose to use. Although it is an undocumented part and all that is available is the pin-out information, I implied that it would work using the static drive technique because it has a single common backplane signal COM. I took a chance that it would work at 3 V - and it did. Actually this LCD turned out to be a better deal than it appeared for the price. The display contrast is high and the viewing angle very accommodating. I even got it to display some alphabetic characters whenever it could be done so that the result is recognizable as an alphanumeric character but sorry no X’s V’s or W’s are possible.



7 Segment LCD Drive circuit

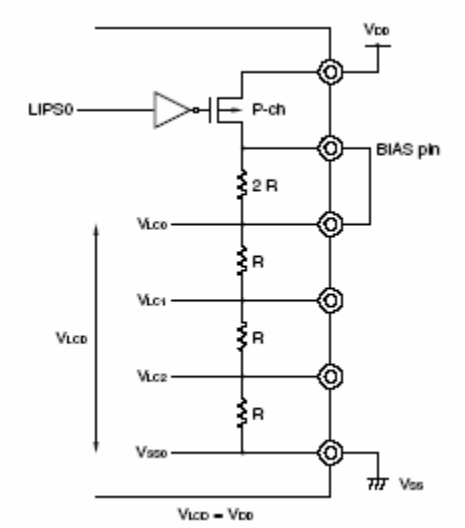
The static mode drive circuit for this LCD is as simple as things can get as shown below. It is a straight 1:1 connection of the S0-S27 signals to the LCD digit segments.



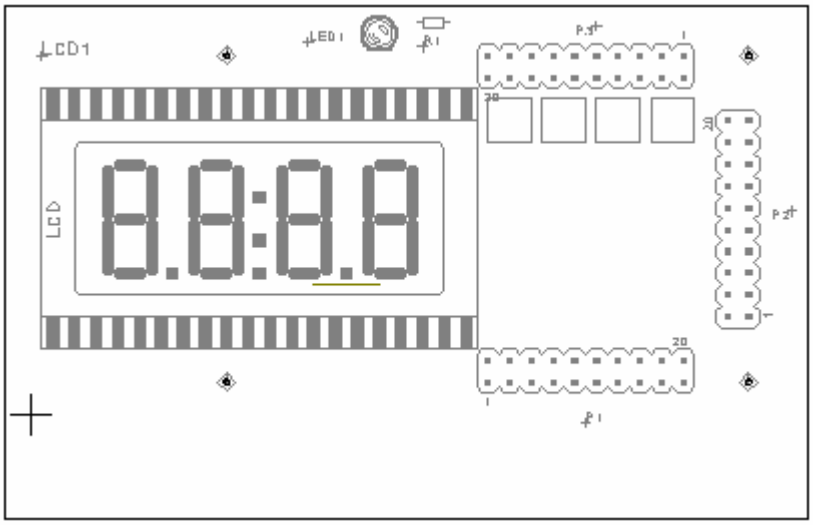
The LCD is mounted on a plug-in board to the KORE9418. On the same board is mounted the debug LED driven by P5.0.

The BIAS pin connection for the static drive mode.

The BIAS pin is connected as follows and as suggested in the upd78F9418 user manual.



Below is the printed circuit board layout for the LCD plug-in board to the KORE9418. Routing those 28-segment signals onto a kitchen-made single layer printed circuit board was not easy and I ended up with 15 jumpers. The result is shown below:



LCD Driver Software

Here is a program that shows how the LCD is driven in software. It displays the date, the time and the year. It repeats this cycle every 10 seconds.

```
[code removed for publication]
```

The LCD display module is shown next.

```
[code removed for publication]
```

Here is a simple electronic trick to have the full 4-digits in static mode while still being able to drive the decimal point and the colon symbol. This trick is based on the fact that an exclusive-or gate can be used as a “controlled” inverter. Below is the truth table for a two input exclusive-or gate:

A	**B**	**A Exclusive OR B**
0	0	0
0	1	1
1	0	1
1	1	0

The truth table for the exclusive-or logic function is easy to remember. Whenever the inputs are different, the output is a 1 otherwise it is a 0. Now look at it slightly differently in two separate groups.

Non-inverting group

Regarding the A input as a ‘control’ input, observe that when A = 0 the output follows B.

A	B	A Exclusive OR B
0	0	0
0	1	1

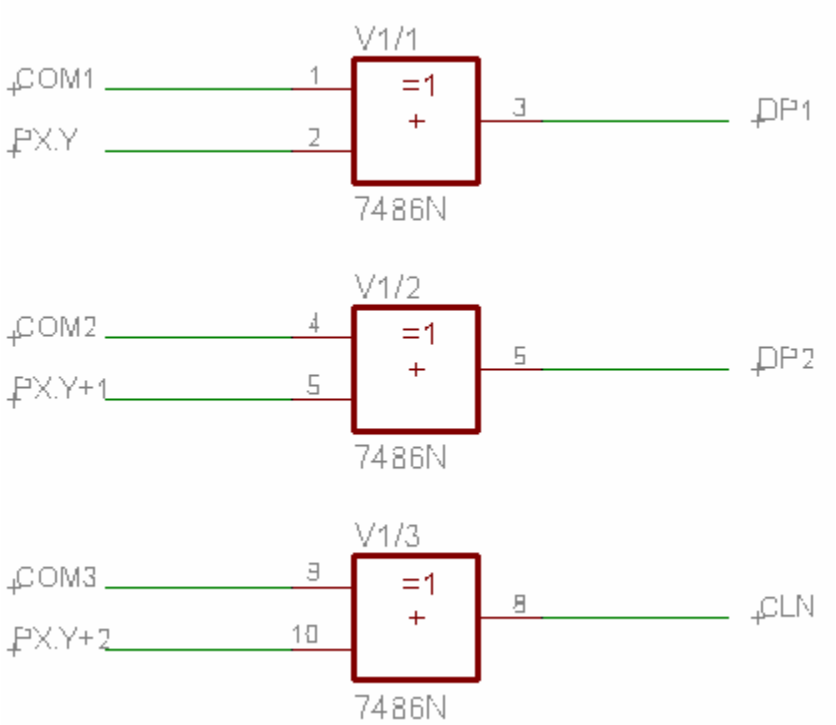
Inverting group

Again, regarding the A input as a ‘control’ input, when A = 1 the output is the inverse of B.

A	B	A Exclusive OR B
1	0	1
1	1	0

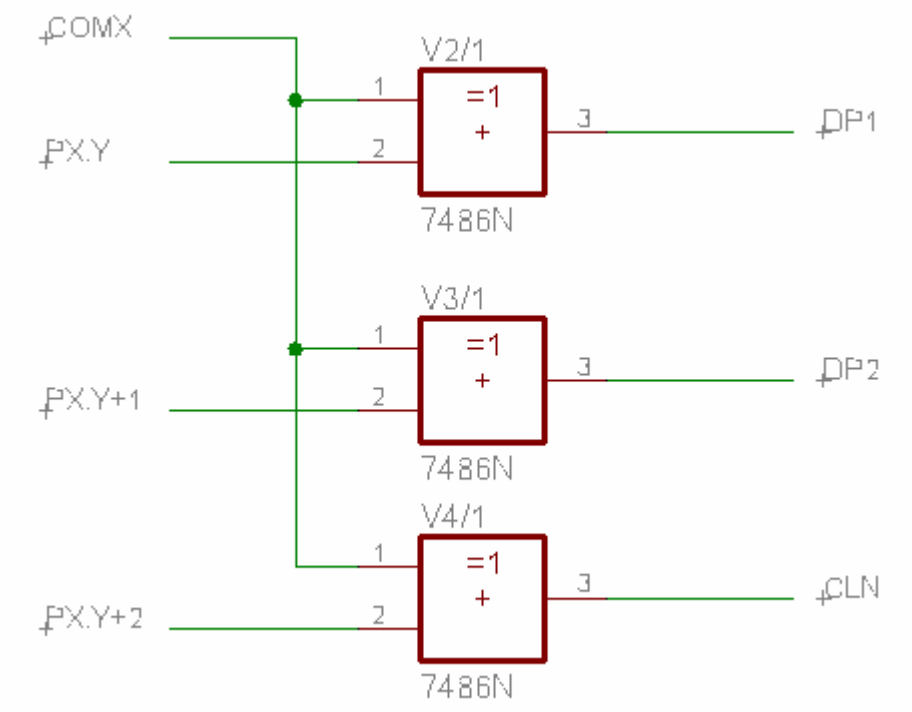
Thus an exclusive-or gate can be used as a controlled inverter. Simply regard the one input as the control input. When it’s a 0 the output is non-inverting. When it is a 1, the output is the inversion of the other variable input. This property of the exclusive-or gate can be put to good use to fake out additional segment signals from the COM0 to COM3 signals in the static mode. In the static mode all the COM signals are the same. A single COM signal can be used to drive the LCD backplane common signal whereas the COM1 to COM3 signals can be used individually to drive the decimal points and colon symbols on a regular 4-digit LCD.

The following circuit will adequately do the trick:

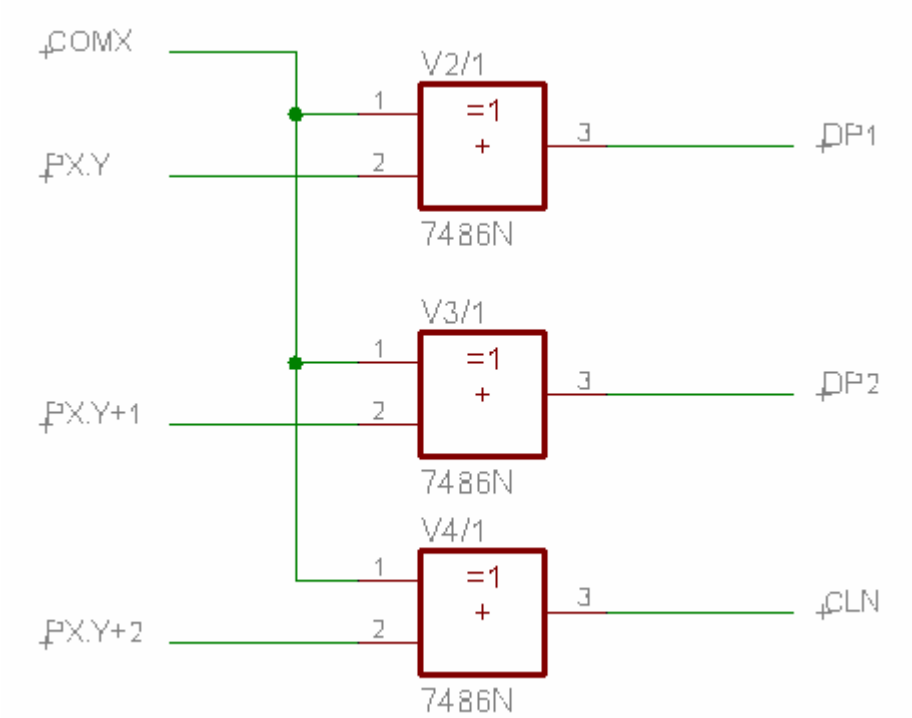


Assume that PX.Y..PX.Y+2 are port outputs. Pull-ups may be required.

This one is even simpler and uses less PCB traces but places a larger fan-out requirement on the COM signals.



I did not implement this trick on the LCD plug-in board the first time around when I had the chance. I regret this decision now. Texas Instruments makes single three-input exclusive-or gates in a small outline package. They operate down to 3 V. This is the SN74LVC1G386. If time permits, I will at least implement a single decimal point as a tucked-on wire-wrapped piece of glue logic. I have developed a craving for decimal points.



I did not implement this trick on the LCD plug-in board the first time around when I had the chance. I regret this decision now. Texas Instruments makes single three-input exclusive-or gates in a small outline package. They operate down to 3 V. This is the SN74LVC1G386. If time permits, I will at least implement a single decimal point as a tucked-on wire-wrapped piece of glue logic. I have developed a craving for decimal points.

The power section

The SmartPlanter is designed to be autonomous in the sense that it does not require an AC power source to operate. Power safety regulations are costly to gain certification for and besides it seems too cumbersome to have each planter connected to the mains 120 VAC supply.

I could have used a mains step down transformer or adapter to feed a low level DC power supply to each planter, but then again, the wiring would make it look unsightly and would reduce the mobility of the planter.

Hence, the SmartPlanter somehow had to be self-powered. The only obvious option is to use batteries, but having to replace batteries frequently is a pain.

Fortunately one of the features of the upd78F9418 is its ability to switch to a lower rate clock to conserve power, but still, the SmartPlanter needs to operate a water pump and this takes enough power to drain a battery in a relatively short amount of time.

The idea used here is to complement the power requirement using solar panels. It makes sense to use solar panels in this application because plants cannot survive without sunshine. Wherever there are plants there is bound to have at the very least some solar energy. Even if the SmartPlanter is installed indoors it would be located in a place where the plant receives at least a few hours of sunshine every day even if it is diffused and not direct sunlight. It therefore makes sense to use a solar panel to power the SmartPlanter but the planter should not rely completely on the availability of solar power. A combination of battery and solar power is therefore a reasonable solution

I did not want to have to use a solar panel that is too large because it would interfere with the ergonomics of the planter. Whichever solar panel is to be used has to be small, discrete and fit nicely within the planter.

I found high-efficiency solar panels that are a mere 36 x 36 mm capable of delivering high, short circuit currents and an open circuit voltage of 5 V or more in direct sunlight. These are perfect for trickle charging 3.6 V Lithium Ion batteries. My solution is based on the use of this type of solar panel charging common 3.6 V Lithium Ion batteries. This way the batteries will last a while before having to be replaced.

For the solar panels I used two Panasonic BP 24-2221 solar panels in parallel. The solar cells are 37 x 33 mm in size. They provide 5.5 volts (open circuit) at 15mA (short circuit)

in noon sunlight. These are the SC3733 - 37 x 33mm Panasonic Sunceram Solar Cell from SolarRobotics LTD.

For the battery I used a single 3.6 V 1050 mAH Lithium Ion rechargeable battery made by A&TB Japan. These were purchased (on sale) for $1 each.

Two of these solar panels in parallel should deliver at least 20 mA in moderate sunlight which is still not enough to drive a water pump motor. Even those tiny motors found in ZipZap remote control cars draw around 40 mA. I did not want to drive the pump motor from the 3.6 V batteries because they will run it dry too early. Examining the circuitry found in ZipZap RC cars closely gave me the answer - supercapacitors!

Supercapacitors

Supercapacitors are high value capacitance capacitors. Capacitance values are in the range of 1F to 10F and recently I have seen some in the 100F range in EDN. These capacitors are amazing in the fact that they can achieve such high capacitance values in such small sizes. They have limited voltage ratings however and care must be taken in selecting them although they can sustain momentary overvoltages. The most common types are found in the 2.5 V range with others in the 5.5 V range. I picked a supercapacitor with a 5.5 V rating to match the solar panels.

What makes supercapacitors more attractive than batteries when short bursts of high current are required is the fact that they have very low equivalent series resistances (ESR), of the order of milli-ohms. They can generally deliver much higher currents than batteries of similar size can deliver. The other interesting aspect is that they can be recharged in a matter of seconds and not hours like batteries and in addition they have extremely long life for they can be charged and discharged practically indefinitely for they have no memory effect like rechargeable batteries. All in all, it is the perfect choice to power a watering pump.

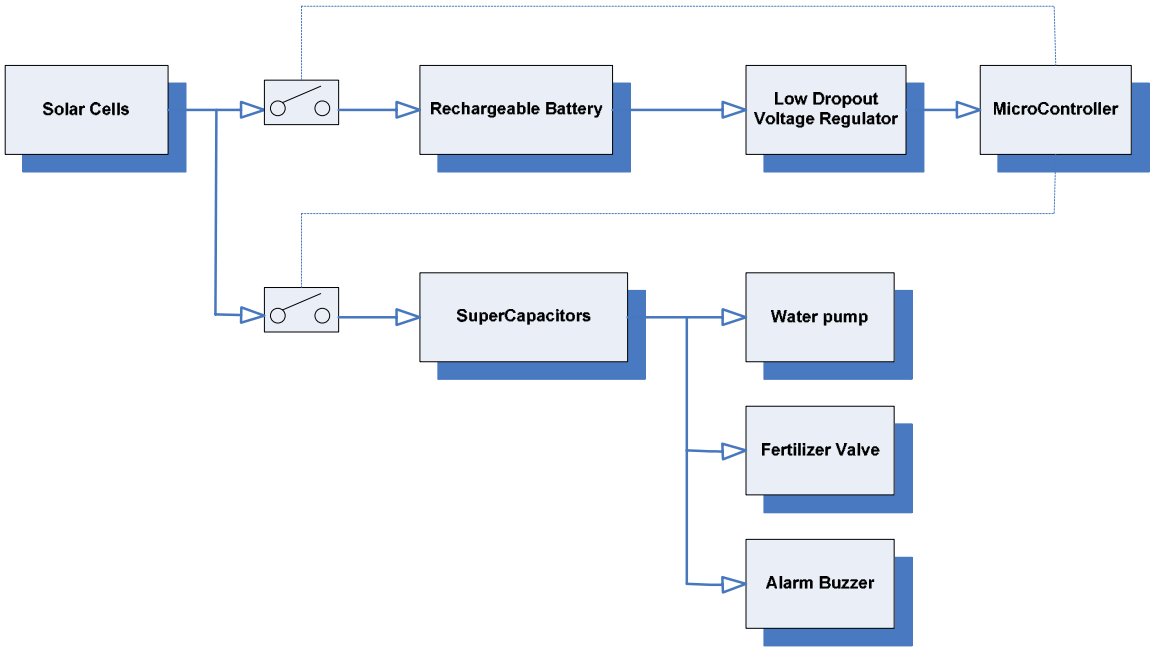
In this design two supercapacitors of the Maxcap type of 0.47 F and rated at 5.5 V are used connected in parallel. These supercapacitors are charged by the solar cells. They are then used to drive the watering pump. The battery only supplies power to the microcontroller. The software controls whether the solar panels charge the batteries or the supercapacitors, when sunlight is available. The software performs power cycling, evenly distributing the solar energy to charge the batteries and the supercapacitors.

In this design the combination of the following features help make the SmartPlanter completely autonomous:

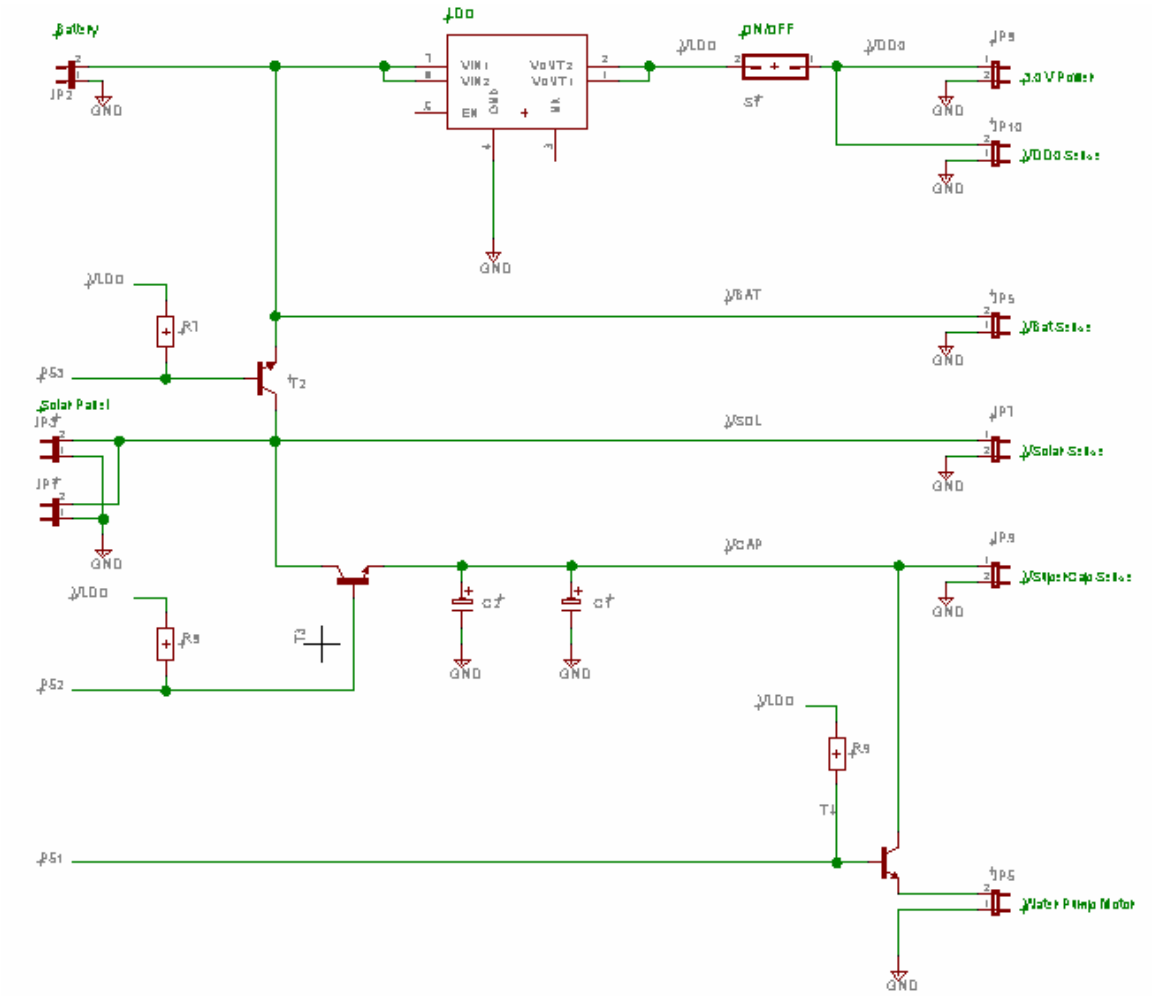
- Power cycling solar energy to the battery and supercapacitors
- Using the supercapacitors to power the higher current devices such as the watering pump
- Using the battery only to power the microcontroller
- Using the power saving features of the microcontroller and switching between the high speed 4.9152 MHz and 32 KHz clocks

- Using the built in AD converter to measure voltages and make decisions based on the readings. All voltages are measured: solar cell voltage when open circuit, solar cell voltage when charging the battery, solar cell voltage when charging the supercapacitor, supercapacitor voltage, battery voltage, microcontroller VDD voltage.

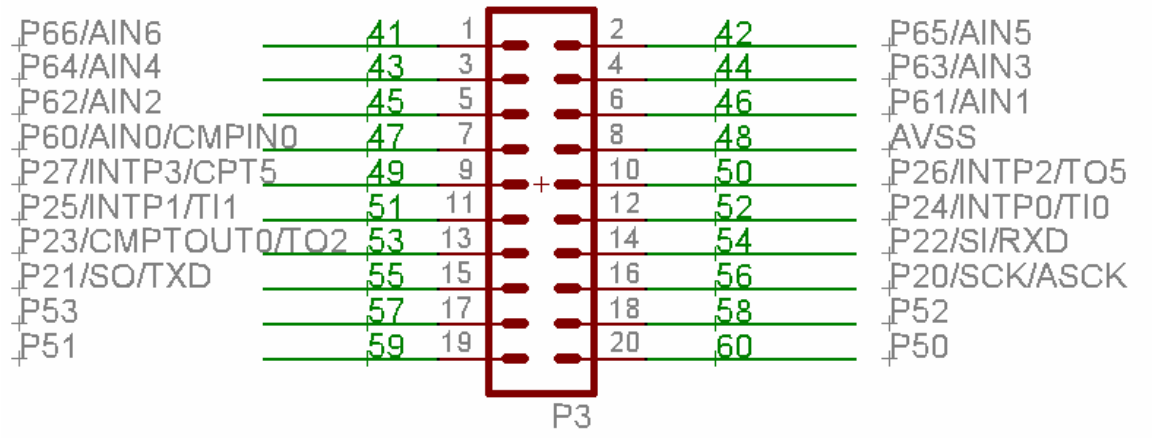
A block diagram of the SmartPlanter power section is shown below:



The circuitry implementing the above block diagram is shown below:



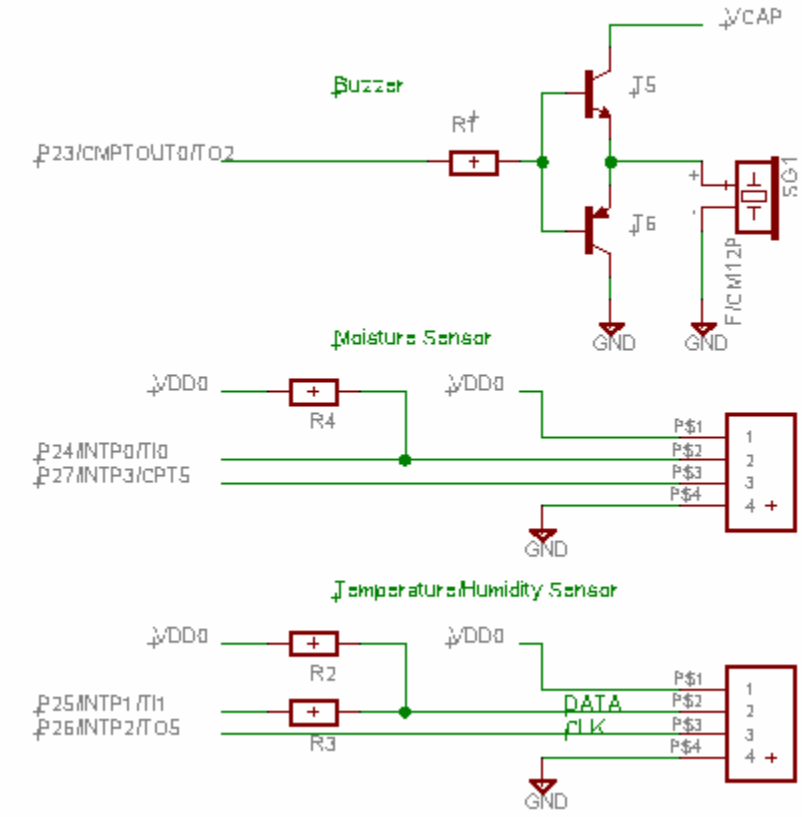
This circuit is implemented as a single layer printed circuit board which is a plug-in addition to the KORE9418. It picks up the following signals from the KORE9418 connector.



The following circuitry also fits on this “Power Section” board.

Alarm Buzzer circuit

A high efficiency piezoelectric buzzer driver circuit is powered from the supercapacitor and driven by the T02 output. Timer 02 is the only timer that is capable of generating square waves of programmed frequencies with zero software overhead that is all using the internal hardware. It was chosen to drive the buzzer for this reason. It uses the 32 KHz external crystal because it is intended to be generating audible signals in the hundred of Hertz to low kilo Hertz range. This audible alarm is used to raise a low water-level alarm. A piezoelectric buzzer was chosen because of its low current consumption. Experiments have shown that even with just above a volt charged on the supercapacitor, the buzzer can go on making a noise for quite a long time. This is exactly what is required in this application. Since water is crucial for the survival of the plant, the SmartPlanter makes an annoying noise to attract attention when it senses that the water level is reaching dangerously low levels. The buzzer is also used to provide audible feedback when a keypad key is pressed.



Moisture sensor interface

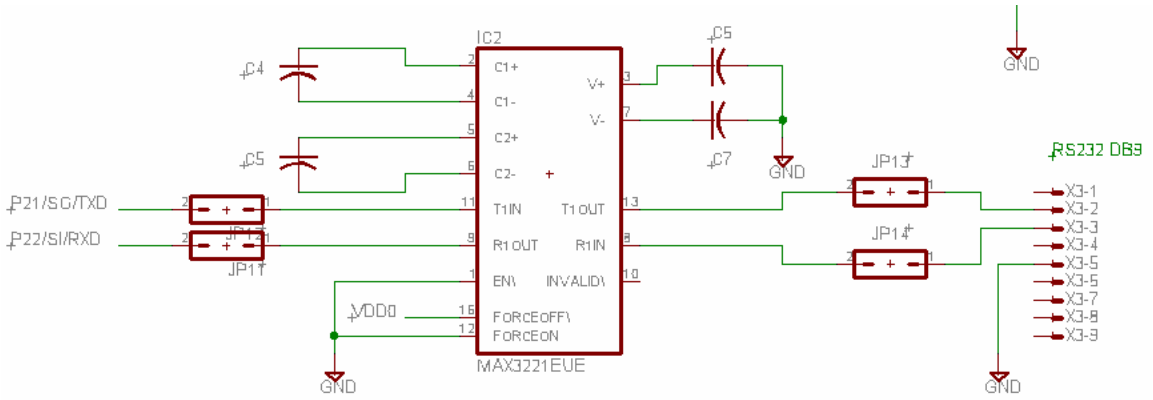
Below the buzzer driver circuitry is the header feeding the signals to the moisture sensor measuring circuit. This sensor is crucial to the operation of the SmartPlanter.

Temperature and Humidity sensor interface

Following this is a header feeding the signals driving the SHT11, a humidity and temperature sensor. This circuitry is optional and only used if evapotranspiration estimates are to be performed and used.

RS232

In addition, for research purposes it is highly desirable to provide the ability for one to log the variables measured and controlled by the SmartPlanter, for this reason a host interface is essential. Provision for this is made by the addition of an RS232 interface. This circuitry is optional but provides the ability to perform data logging.

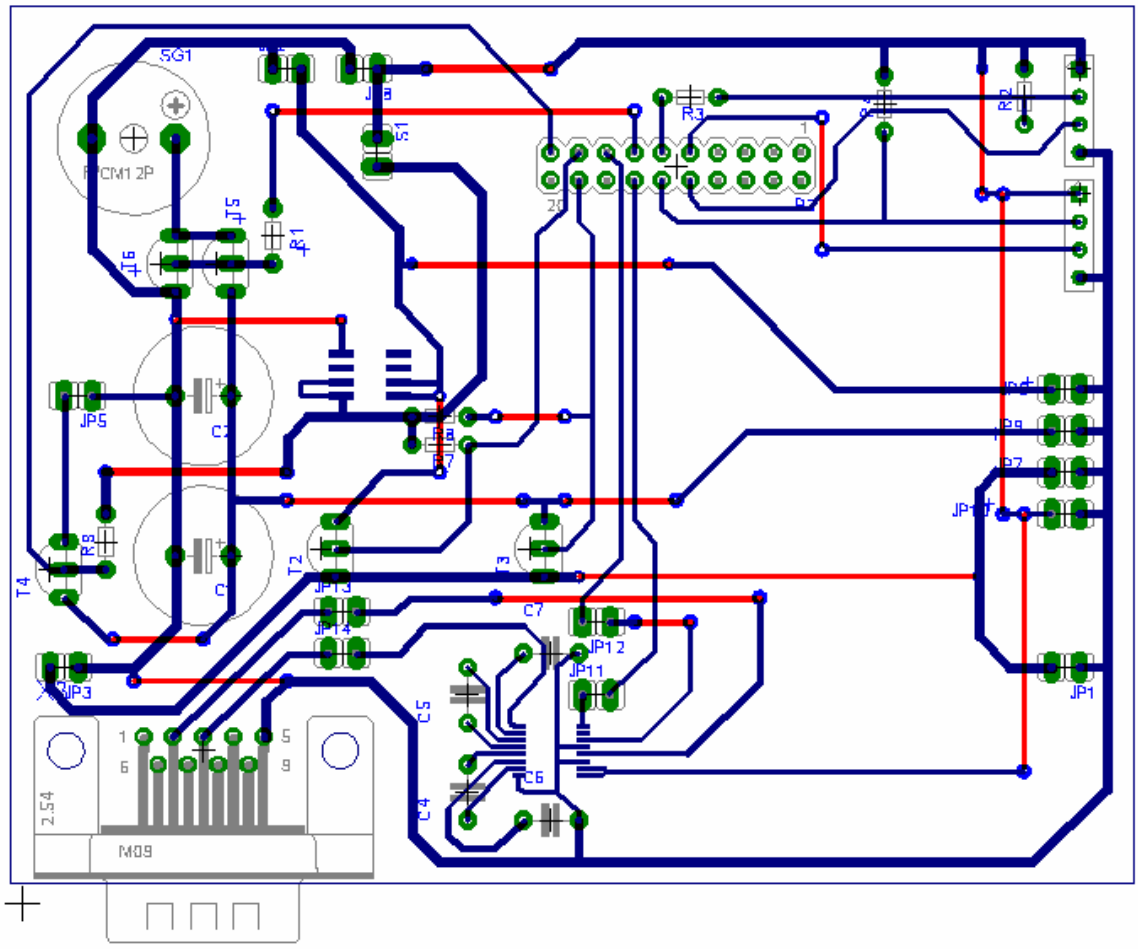


The jumpers on the RS232 level side are used to correct any TXD/RXD mismatch according to the host interface connector used. The jumpers on the logic level side are to facilitate the SmartPlanter,s software development. When it is connected to the M-Station these jumpers should be removed so as not to interfere with the M-Station signals. The MAX32221 operates at 3 V and has a power-saving built in feature. When it senses that there are no RS232 level signals present on its input, it automatically shuts itself down to conserve power.

It is planned that this interface will be connected in future to one of several other interface types namely:

- IrDA
- USB using an RS232 to USB converter
- RF Transceiver for a remote wireless interface

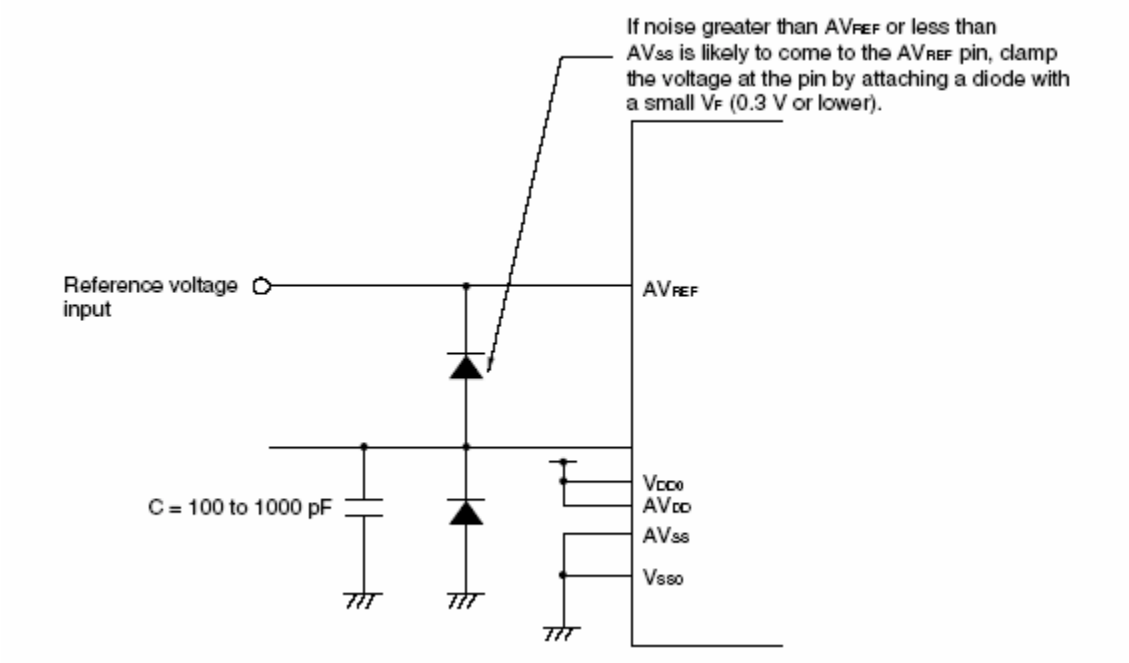
Here is the printed circuit board implementation of the above circuitry, all handcrafted in the kitchen sink:



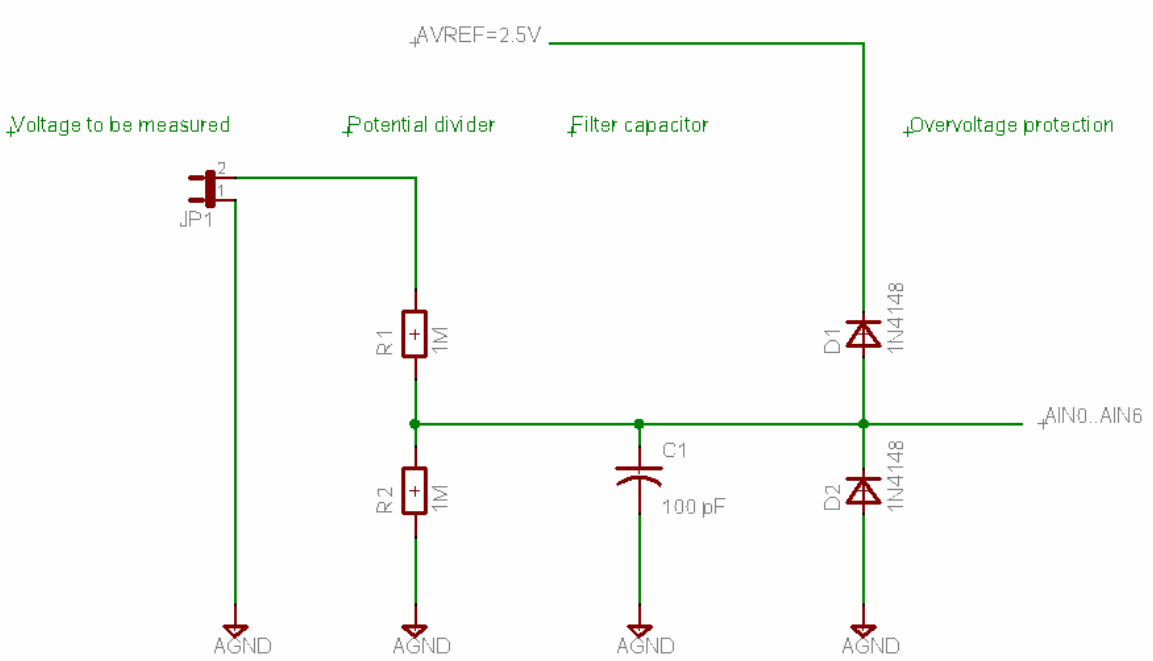
The headers on the right hand side of the board are the voltage sensing headers and feed into the analog signal processing board which is the next topic of discussion.

The Analog to Digital converter

The 10-bit A/D converter feature of the upd78F9418 is essential to this application. It is used to perform power management and actuators control. The upd78F9418 user manual recommends the following protection circuitry on the analog inputs which I have implemented.

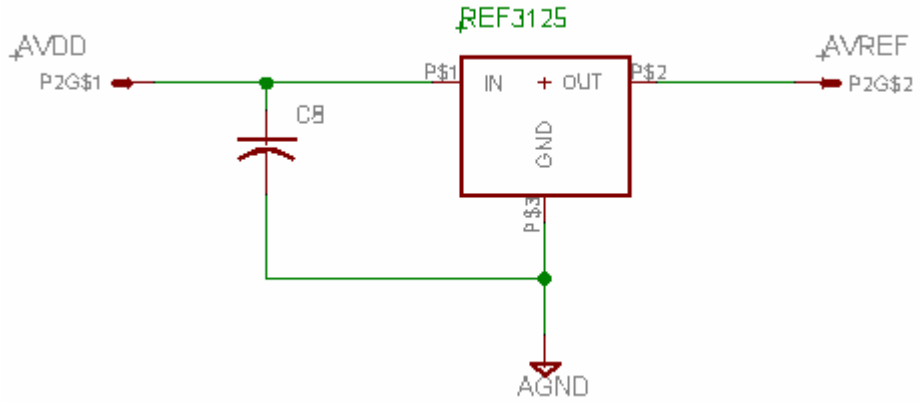

If noise greater than AVREF or less than AVSS is likely to come to the AVREF pin, clamp the voltage at the pin by attaching a diode with a small VF (0.3 V or lower).
Reference voltage input
AVREF
C = 100 to 1000 pF
VDD0
AVDD
AVSS
VSS0

All my measurements are ratio metric and make use of a potential divider. The voltages to be measured include the solar cell voltage and supercapacitor voltage. The solar cell voltage may rise as high as 5V under direct sunlight conditions and this voltage source charges the supercapacitors which are rated at 5.5V. The recommended value of AV_{REF} is 2.5V, which is about half of the voltages that I am trying to measure. I used a simple potential divider to halve the voltages fed to the A/D inputs. I compensate for it in software using a multiplying factor of 2. The A/D input circuit is as shown below:

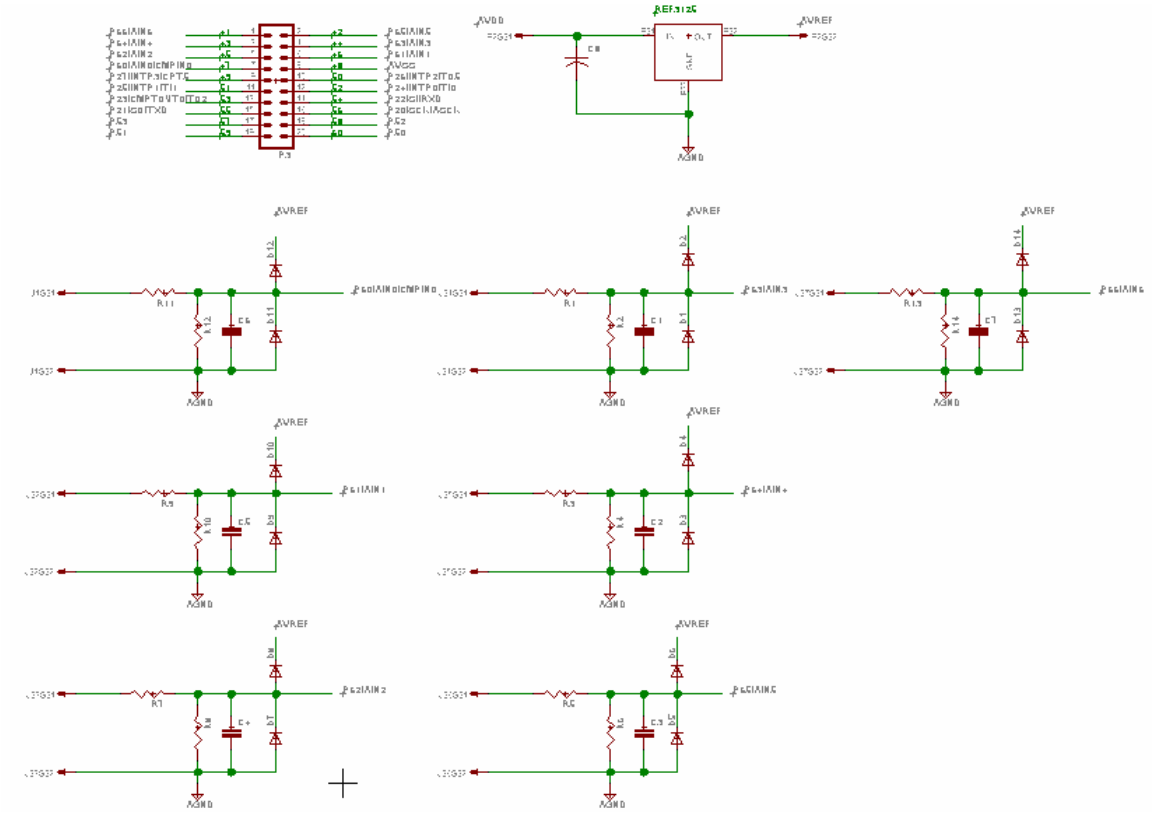


This input circuitry is probably overkill for this application for it is battery operated and is free of mains hum, besides I did not expect the input to be noisy. Experience has shown that solar cells have a lot of high frequency variations. The idea was to implement a somewhat generic A/D converter signal conditioning add-in board for the KORE9418 that could be used in other applications. Although the printed circuit board implementation caters for the circuit above, components can be omitted to simplify things whenever needed but at the very least, for this application the potential divider section is required.

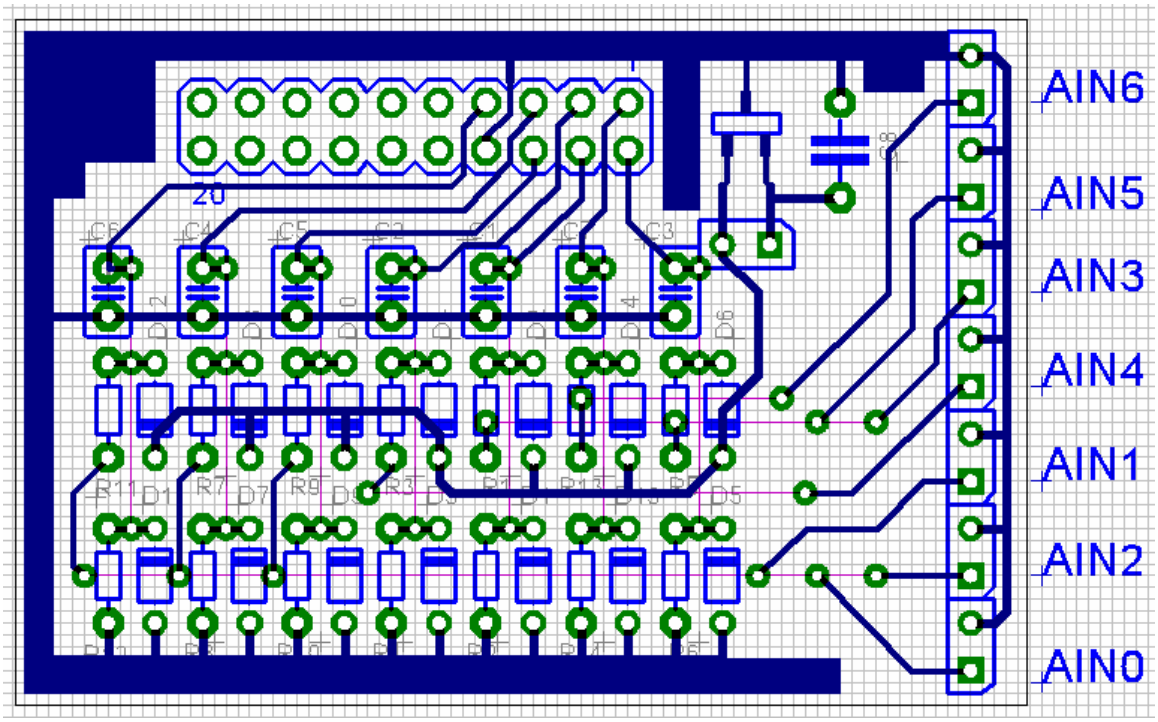
The microcontroller operates under a stable 3.0 V provided by a LDO regulator as seen in the power section of this document. The chosen 2.5 V reference is the REF3125, a Burr Brown product from Texas Instruments. It was chosen for its low quiescent current of 100 µA, its low dropout of 5 mV and its 20 ppm/◦C drift performance. In addition an output capacitor is not required. In this low noise application the recommended input capacitor was omitted without any effect on performance.



Here is the complete circuit diagram for the analog input plug-in board:



The single-layer printed circuit board implementation is shown next. Soldering the smd voltage reference by hand was not funny.



The A/D software

Although the A/D can operate continuously and can generate interrupts, it is used in polled mode because this application does not require frequent measurements. Measurements are taken on demand and the A/D conversion complete flag is polled.

The following program measures the voltage on the solar cells under three distinct conditions:

1. When the solar cell is open circuit. The open circuit voltage can be used as a measure of solar irradiation. This parameter is required in the evapotranspiration equations.
2. When the solar cell is charging the battery. This is used as a check that battery charging is actually taking place. This voltage should be lower than the open circuit voltage.
3. When the solar cell is charging the supercapacitors. The supercapacitors present a load to the solar cell and this voltage is lower compared to the open circuit voltage.

The program also measures the voltage on the battery and on the supercapacitors respectively. The results are sent to the serial port. It is interesting watch how these voltages vary when the battery and super capacitors are being charged. Supercapacitors charge very quickly compared to the battery. Their voltage approaches the solar cell voltage under load conditions in a matter of 20 to 30 seconds. This program does not use the analog input board described above for it was written prior to the latter being constructed. The only difference is that in this case AVREF = VDD, AVDD = VDD, VDD=5V.

```
[code removed for publication]
```

The header files used above are:

IO.h – I use macros for simple things as configuring ports and toggling their outputs. These do not warrant the code space used by making these functions and they run faster.

```
[code removed for publication]
```

AD.h follows:

```
[code removed for publication]
```

AD.c

```
[code removed for publication]
```

Here is a more elaborate program that displays the results on the measurements on the LCD using a combination of alphabetic (whenever possible) and numeric characters. This program uses the analog input board printed circuit board. The difference is that AVREF=2.5V, the gain (potential divider) is still 2.0.

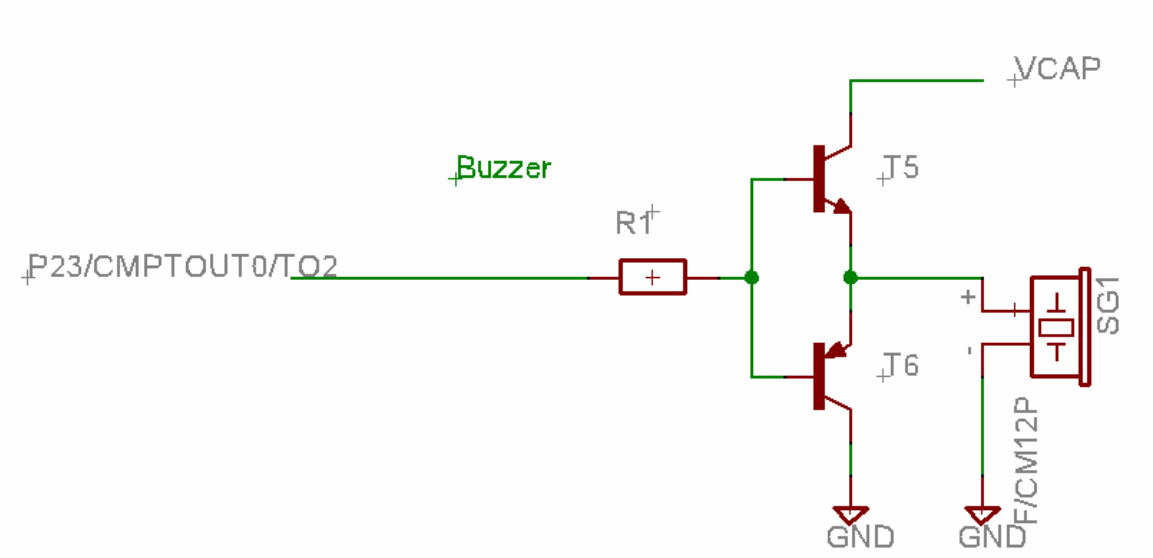
```
[code removed for publication]
```

This program displays the following on the LCD

LCD Display	Voltage source/Description
SOLO	Solar cell open circuit
SOLC	Solar cell charging supercapacitor
SOLB	Solar cell charging battery
SCAP	Supercapacitor
BATT	Battery
H2OL	Water level
FERL	Fertilizer level
AI05	Voltage on AIN05 (spare)
AI06	Voltage on AIN06 (spare)
TOD	Time of day
DATE	Date
YEAR	Year
8888	LCD segment test pattern

The alarm/buzzer software

I used the following circuit to drive a piezoelectric buzzer to serve as an audible alarm to raise the alert when the water level in the SmartPlanter is reaching low levels. This is meant to prompt users to refill it. Depending on the plant species, this water reservoir should last about 3 to 4 weeks. That is enough to safely go on a vacation without worrying about losing plants due to under-watering for indoor plants. For outdoor plants the SmartPlanter is designed to collect rain water in its reservoir. This further extends the period of time that the plant can be unattended and the reservoir topped off.



I used the P23/CMPTOUT0/TO2 signal to drive the buzzer because T02 can generate square waves with no software overhead. Hence it does not require bit banging a port to generate an audible square wave. Without this feature I would probably have used a timer or implemented an oscillator around the buzzer circuit. Here is the software that drives the buzzer.

```
[code removed for publication]
```

To use the buzzer use the following code:

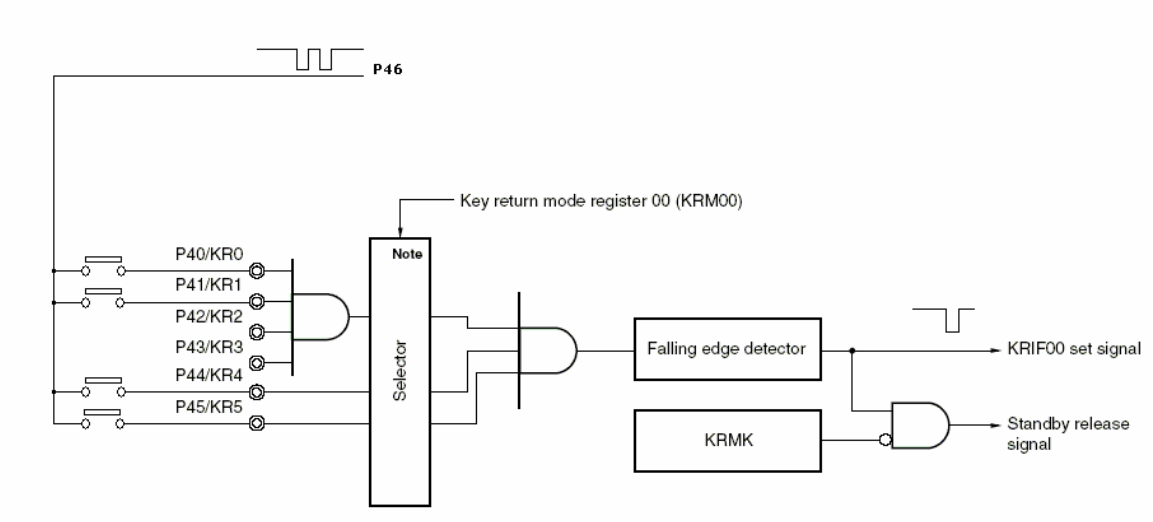
```
[code removed for publication]
```

To have the ability to change the buzzer tone frequency on the fly without having to turn it on again, simply change the SetBuzzerFrequency() macro to:

```
[code removed for publication]
```

In other words just change the CR value on the fly without turning T02 off. A note in the upd78F9418 recommends turning T02 off before setting CR because it may fire spurious T02 interrupts but in this application we don't care about this because the T02's interrupt mask is always off.

Keypad Block Diagram



What is Missing?

I had been working on this SmartPlanter idea as a viable product long before the Mad Design Contest came along. The NEC's Mad Design Contest provided the wonderful opportunity of not only having my ideas made public but giving me access to the design tools, the software and the hardware. The NEC's upd78 series of microcontrollers seemed like a good fit for this application therefore I decided to participate. I knew all along that this project would require considerable research and that I may not finish it in time.

One of the things that is not visible or aspects of this project that I am not stressing has to do with the mechanical design aspects. I researched the market extensively for a low cost pump that works on 3 V or less and there simply isn't any. If there is one out there that I missed, chances are it works on 12 V or more and is some kind of high end medical device and costs a fortune. This led me to designing a low power pump from scratch. For this purpose I purchased a lathe and a milling machine and experimented with various ideas. The pump that is included in this submission represents the first working prototype of such a pump. I am delighted with its performance even as an early prototype but there is still some room for improvement. It is low cost, completely submersible, chemical-resistant and works from a mere 4 cm^2 of solar cell. I would appreciate that its design be kept low key and that it not be reverse-engineered.

The measurement of moisture using Gypsum blocks is another topic that took considerable research and I am still working on calibration techniques. Whatever information is included in this submission is to be considered a work in progress.

The topic of Evapotranspiration (ET), its adaptation and application for this particular use represents a considerable investment in IP. For this reason at the 11^{th} hour and upon recommendation I choose to actually not present this as part of the solution. I hand planned to include it as part of the project in the form of a compiled library but finally refrained from doing so. In principle this should not affect the perceived usefulness of the the SmartPlanter and should be considered a refinement. The moisture sensor tells the SmartPlanter that the soil is getting dry and it needs to water. It is the ET equations that tell the SmartPlanter how much water to dispense based on estimates of how much water was lost the day before. Well this part is simply not included for reasons outlined above. This means that the SmartPlanter in its current state waters fixed amounts. To estimate ET several parameters are required namely the atmospheric pressure, the minimum, maximum and daily temperatures, humidity and dew point, and wind speed. Atmospheric pressure can be estimated as a function of altitude which can be derived from geographical location. Wind speeds can be assumed constant for indoor plants and being within a certain range for outdoor plants. This project and the printed circuit board make the assumption that temperature, humidity and dew point measurements are to be made using the SHT11 humidity and temperature sensor from Sensirion. The PCB traces and circuit diagrams allocate a CLK and DATA signal to a header to drive a SHT11 device. The upd78F9418 synchronous serial port mode does not work with this device because the latter is somewhat non-standard. The software has been tested on the DS92C420 but time did not permit to actually integrate it within this application.
Plant moisture control is a slow process. I thought that it would be difficult to see visible effects on the short run. It would take days of observation to see what the SmartPlanter is doing and actual statistical field trials are still to be done to characterize plant species and develop "recipes" that is watering/fertilizing schedules for them. This work is likely to be a long process that is beginning this year. I therefore realized that it would be difficult to judge hence my submission essentially shows how I intend to use the features of the upd78F9418 to achieve these goals. My submission therefore attempts to demonstrate this as much as possible in that it collects all the key measurements required with the idea that, in principle, these can be used to make decisions about plant growth.

I never had time to implement at least one of those decimal point driving tricks. For me who has a working knowledge of the system it is easy to imply the location of the decimal point but for others the display may be somewhat cryptic.

All in all this submission in my opinion represents a good effort of what can be achieved in a relatively short period of time. A lot of work is still required before the SmartPlanter can make its claims. When I reach this point, will I still use the NEC microcontroller? It makes sense because I found it mostly lacking in no way for this particular application. I most likely will switch to a microcontroller with a graphics LCD display. I had no issue with the tools provided and generally things worked the first time around for whatever it is that I set out to make the chip do. I have definitely enjoyed and gained from this experience.

Final shipped SmartPlanter demo program

```
[code removed for publication]
```